9. Real-Time Services

9.1 Overview

The real-time services discussed in this section include: logical string processing, Telemetry Service, Commanding Service, and Ground Telemetry Processing. Logical string processing services provide FOT and IST users the ability to employ real-time services in order to monitor and control real-time, simulation, and historical replay activities for a given spacecraft. The Telemetry Service provides the ability to monitor spacecraft telemetry associated with real-time, simulation, and historical replay activities. The Command Service provides the ability to send scheduled and real-time commands to the spacecraft as part of a real-time logical string, and to a simulated spacecraft as part of a simulation logical string. The Ground Telemetry Processing Service provides the ability to monitor the Network Control Center (NCC) and EDOS external interfaces as part of a real-time logical string. Historical replay capabilities are also provided using archived external interface data as part of a replay logical string.

Real-time services are driven by ECL directives. ECL directives invoking and customizing real-time services can be manually entered via the command line on the Control window. Alternatively, real-time ECL directives can be entered into the system through automated means, such as command procedures, ground procedures, and ground scripts. For a discussion of the various mechanisms by which directives are entered into the system, refer to Section 6. This section provides the following information for each ECL directive used to invoke real-time services:

a. A description of the keywords associated with a directive.

b. Valid mechanisms for entering a directive into the system.

c. Event messages that can be expected after entering a directive.

d. The database version associated with the logical string (i.e., 1.0).

Additional information regarding ECL directives, their usage, event definitions, and recommended responses may be found in Appendices A and B.

9.2 Logical String Processing

A logical string can be defined as the hardware hosts and the application software supporting a real-time, simulation or historical replay activity for a given spacecraft. An activity in this context can be described in terms of three attributes that collectively and uniquely identify each logical string established by the system. The three attributes include:

a. An identifier for the specific EOS spacecraft being supported (e.g., AM-1).

b. The source of the data being processed (i.e., real-time, simulation, replay).

c. The intended use of the logical string (i.e., operational, test, training).

Logical strings employ shared or dedicated resources and are either globally visible or visible only on the userstation establishing the string. A shared logical string is established with an EOC Real-Time server as its host and is globally visible to users within the EOC and users connected to the EOC through ISTs. A dedicated logical string is established on a userstation at the EOC or at an IST, and is dedicated to and accessible by a single user. Some capabilities accessed via connection to a shared logical string include:

a. Operational monitoring of a real-time contact with a given EOS spacecraft.

b. Simulation of real-time contacts with a given EOS spacecraft for training purposes.

c Replay of historical spacecraft telemetry from a given EOS spacecraft to support the testing of new release software.

Shared logical strings are created to support real-time, simulated, and historical replay activities because they concurrently provide multiple EOC and IST users access to the same data streams. Only users at the EOC with ground control privileges may establish shared logical strings. Dedicated logical strings support off-line activities, such as historical replays and analysis requests and may be created from an IST or from the EOC by any user.

Logical strings are identified by a numeric identifier ranging from 1 to 999. The identifier for a shared logical string is unique; for example, shared logical string 100 will be the same on every userstation within the EOC as well as every userstation at an IST. A dedicated logical string is identified by a numeric identifier less than 100. Since dedicated logical strings are visible only on the local userstation, logical string 002 on one userstation is not related in any way to logical string 002 on another userstation. Figure 9.2-1 illustrates a sample dynamic page containing logical string configuration parameters.

Logical string 100 in Figure 9.2-1 is an example of a real-time operational logical string. This type of string is unique in that it consists of two strands. One strand of software processes supports the operational spacecraft mission and provides the user access to real-time telemetry data and other command and control capabilities. The second strand represents the backup capability in place in the event of a hardware or software failure that renders the active strand inoperable. The significance of this is further described in paragraph 9.5.

9.2.1 Logical String Processing Directives

ECL directives for real-time logical string processing are derived from either the STRING or TAKE primary keyword. Off-line analysis requests result in the creation of a logical string for historical telemetry processing without the STRING or TAKE directive. For these requests, the directive acting as the catalyst is issued by the Analysis Request Builder. Additional detail concerning the Analysis Request Builder is available in paragraph 10.1.2.2.

The STRING directive has associated secondary keywords for creation and configuration of a logical string, connection to an existing logical string, and transfer of control of mission-critical processing from a string's active processing to its backup processing. The TAKE directive has two associated secondary keywords specifying the privileges that will be requested to perform mission-critical functions within the EOC.

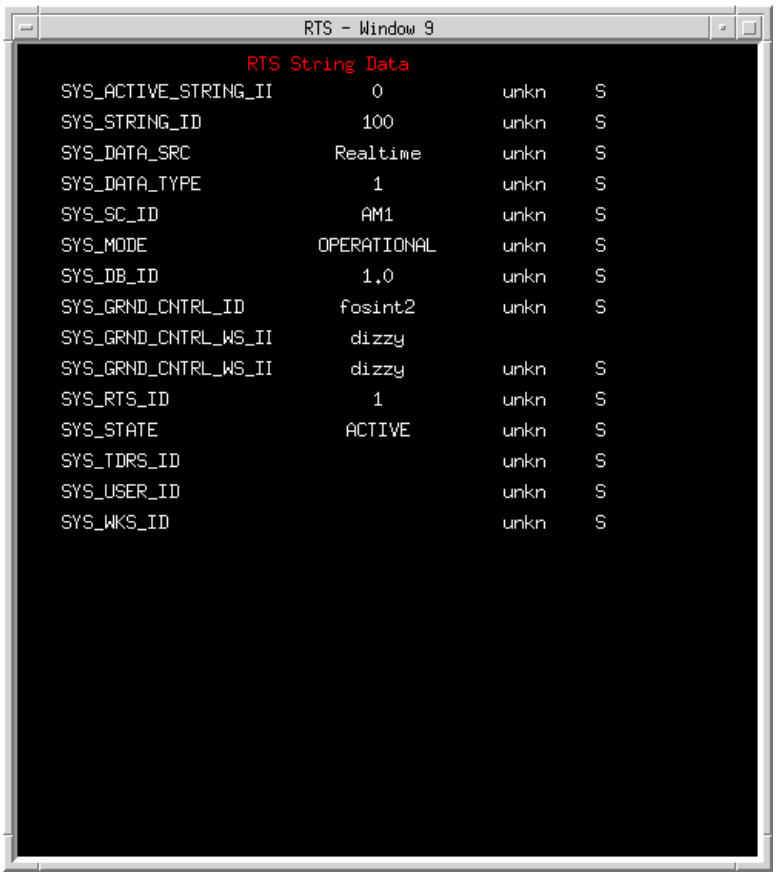

Figure 9.2-1. RTS String Data Display

9.2.2 Logical String Creation

Logical strings can be established manually through a directive or by default when the Real-Time server startup script is run. Nominally, logical strings are established manually. Default logical string creation is used only in conjunction with carefully established and maintained operational procedures. The following sections discuss manual and default logical string creation in detail.

9.2.2.1 Manual Logical String Creation

Logical strings are established manually by issuing the STRING CREATE directive from the Control window ECL command line or by requesting the off-line analysis of historical telemetry. The creation of logical strings that process historical spacecraft telemetry data for off-line analysis reporting is described in paragraph 10.2.

The STRING CREATE directive requires additional secondary directives that specify the configuration of the software executed as part of the logical string. These directives designate the data source, spacecraft ID, database ID, mode and Real-Time server for the string and are explained in detail in the following text. Figure 9.2.2.1-1 provides a sample STRING CREATE directive with associated subdirectives.

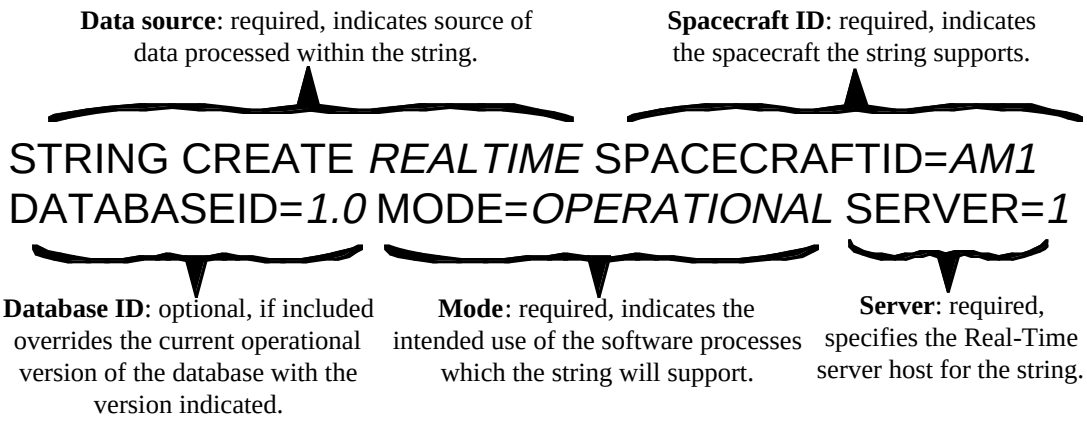

Figure 9.2.2.1-1. Sample STRING CREATE Directive

a. **Data Source**. The type of data being processed within a logical string is referred to as the data source and is required in the STRING CREATE directive. Data processed within a logical string can be from one of three sources: an EOS spacecraft, an EOS spacecraft simulator, or a historical archive. Logical strings that process EOS spacecraft or EOS spacecraft simulator data are established through the STRING CREATE directive using the *REALTIME* or *SIMULATION* data source keywords, respectively.

b. **Spacecraft ID**. The EOS spacecraft ID is required in the STRING CREATE directive. Although the need to specify the spacecraft identifier may seem obvious, the spacecraft identifier is required because logical strings are key components in the simultaneous support of multiple EOS missions.

c. **Database ID**. The DATABASEID keyword and value is used to override the current operational version of the project database and is optional in the STRING CREATE directive. When the database ID is not specified in the directive, the system will establish the logical string using the latest available version of the project database.

d. **Mode**. The mode defines the intended use of software executed by the logical string request and is required in the STRING CREATE directive. Valid logical string modes include: operational, training, and test. The operational mode is specified when the logical string will directly support control center operations. Critical activities such as spacecraft commanding and ground system configuration and control can only be performed through a connection to a logical string which has a real-time data source and an operational mode.

e. **Real-Time server**. The Real-Time server identifier is required information for the STRING CREATE directive. The server keyword allows you to specify the Real-Time server host for the desired logical string. This flexibility enables you to consider the resource loading on each available server when creating a new logical string. In order for backup strings to serve their intended purpose for failure recovery, backup processing for a real-time string must be on a different Real-Time server than the active processing for a corresponding real-time string.

To manually create a real-time, operational logical string via the STRING CREATE directive (assume string identifier is "100"):

1. **Open the Global Event Display window**.

   Click **Tools…** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Open a dynamic page**.

   On the Control window, click **TlmWins…** and select a dynamic page displaying logical string configuration information.

3. **Create a logical string**.

   Enter the STRING CREATE directive in the Control window command line:

   STRING CREATE *REALTIME* SPACECRAFTID=*AM1* DATABASEID=*1.0*

   MODE=*OPERATIONAL* SERVER=*1*

4. **Monitor event messages on the Global Event Display window**.

   Observe the progress of the string creation. Event messages are generated for key software processes spawned during string creation. If a problem occurs during string creation, any processes that were created will be brought down. For each terminated process, an event message will be generated. Table 9.2.2.1-1 identifies the text of event messages associated with string creation or termination and their significance.

5. **Monitor the parameter updates on the dynamic page**.

6. **Connect to a logical string**.

   Once a string has been created, you must connect to the string in a mirrored configuration before assuming ground control. In the Command window's command line enter:

   STRING CONNECT STRING=*100*

   CONFIG=*MIRROR*

Table 9.2.2.1-1. String Creation/Termination Event Message Text

Event Message Text	Significance
Establishing __ service	Initialization of a process.
__ process __ configured	The string creation process has been initialized.
String __ was __	The string has been created.
Shutting down __ process	The process is being shut down.

7. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify that the connection to the desired logical string was successful. If connection is not successful, investigate why by reviewing the event messages generated and repeat step 6 before proceeding.

8. **Assume ground control privilege**.

   Enter the TAKE GROUND CONTROL directive in the command line of the Control window:

   TAKE GROUNDCONTROL STRING=*100*

9. **Monitor event messages on the Global Event Display window**.

   Observe the progress of the request for ground control privilege. If ground control privilege is granted, an event message stating "Ground Control Authority has changed from__to__ for string XXX" will be generated.

9.2.2.2 Backup Logical Strings

In order to facilitate the most expedient and efficient recovery possible from a system hardware or software component failure, the system uses redundant processing executing on separate Real-Time servers. This type of processing operates as a "hot backup" for mission-critical processing provided via logical stings with a real-time data source and operational mode.

In order to take advantage of this redundant processing, a backup logical string must be established for any logical string with a real-time data source and operational mode. Backup logical strings are established manually by an authorized member of the FOT by appending the BACKUP keyword to the STRING CREATE directive:

STRING CREATE *BACKUP* SERVER=*2* STRING = 100

9.2.2.3 Simulation Logical Strings

The STRING CREATE SIMULATION directive establishes a logical string capable of interacting with a spacecraft simulator. A simulation string is very similar to a real-time string except for the number of processes spawned and the data source of the logical string. Fewer processes may be spawned because not all real-time processes are needed to communicate with a spacecraft simulator. For example, Real-Time Contact Management software is used to communicate with EDOS and NCC for AM-1, not the spacecraft simulator. Therefore, it is not spawned for a simulation string.

9.3 Default Logical String Creation

Default logical strings are created by the system when the Real-Time server startup script is run. Operationally, the default logical string creation capability can be used to ensure that a logical string with a real-time data source and operational mode is started as soon as the Real-Time server software is started. The full path name for the file read by the system to create a particular default logical string is:

<release>/<type>/<spacecraft>/DefStringOdf_1.0

where:

<release> = fosa, fosb, etc.

<type>=ops, test, etc

<spacecraft> = am1, pm1, etc.

To disable the creation of a default logical string at Real-Time server startup, the DefStringOdf_1.0 file created during ODF generation can be deleted or replaced with an empty file by the same name. The system will recognize that the file is absent or empty at startup and will not create a logical string. The intended use of this capability will be fully described in operational procedures developed by the ECS FOT.

9.4 Failure Detection

Mission-critical flight operations support hardware and software components are automatically monitored. Status parameters of monitored components may be displayed on a dynamic page. Complement processing is the redundant strand of a local string, e.g. if the local string is a backup strand, the active strand for the same string id is located on a failed RTS. In the event of failure of a real-time operational string component, the following messages detailed in Table 9.4-1 are generated and displayed on the Global Event Display window to guide in proceeding with failure recovery.

9.5 Failure Recovery

The following sections describe failure recovery procedures recommended in the event of hardware or software component failures affecting mission-critical flight operations support. Software executed as part of a real-time, operational logical string in the active state is considered mission-critical. All other types of logical strings executing within the system are for off-line analysis, software testing, or personnel training and are not deemed critical to the life of an EOS spacecraft.

A real-time, operational string is composed of an active and backup strand. The active strand is created manually via the STRING CREATE directive or through the DefStringOdf_1.0 file (refer to paragraphs 9.2.2.1 and 9.3, respectively). Once the active strand is created, the backup strand is created by an authorized member of the FOT via the STRING CREATE BACKUP directive. The backup strand mirrors the active processing of the real-time, operational string and ensures that

Table 9.4-1. Failure Recovery Event Message Text

Event Message Text	Significance
Active strand for string ID _____ failed due to failure of process. Suggest failover	The active strand for the string indicated has failed. Perform failover to the backup strand.
Backup strand for string ID _____ failed due to failure of process. Suggest string delete BACKUP strand	The backup strand for the string indicated has failed. Delete the backup strand.
Realtime server_____, host name_____, failed.	The Real-Time server host indicated failed.
Complement for local string _____unavailable. _____string assigned to failed realtime server _____.	The complement for the local string indicated is unavailable.
Software monitor request failed for string _____. Suggest delete string.	The software monitor request for the string indicated failed. Delete the string.
Realtime server _____, host name _____ , available.	The Real-Time server host indicated is available.

if a Real-Time server fails within the EOC, or if a critical software process executing in an active, mission-critical logical string fails, an operator will be able to recover from the failure in the shortest possible time. Failure to establish backup processing for each real-time, operational string may prevent operations personnel from recovering from a hardware or software failure within the one minute system requirement.

As long as a backup string has been created for the real-time operational string, string failover may be performed via the STRING FAILOVER directive which transfers control of the spacecraft from the active to the backup string. The Ground Controller for a real-time, operational string is the only individual authorized to perform a string failover. The only additional information required for the STRING FAILOVER directive is the STRING ID identifying the failed string.

To perform failover for a real-time operational logical string (assume string identifier is "100"):

1. **Open the Global Event Display window**.

   Click **Tools...** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Connect to a logical string**.

   You must connect to a string in a mirrored configuration before assuming ground control. In the Control window's command line enter:

   STRING CONNECT STRING=*100* CONFIG=*MIRROR*

3. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify that the connection to the desired logical string was successful. If connection is not successful, investigate the reason why by reviewing the event messages generated and repeat step 2 before proceeding.

4. **Assume ground control privilege**.

   Enter the TAKE GROUND CONTROL directive in the command line of the Command Control window:

   TAKE GROUNDCONTROL STRING=*100*

5. **Monitor event messages on the Global Event Display window**.

   Observe the progress of the request for ground control privilege. If ground control privilege is granted, an event message stating "Ground Control Authority has changed from__to__ for string 100" will be generated.

6. **Open a dynamic page**.

   On the Control window, click **TlmWins...** and select a dynamic page displaying logical string configuration information.

7. **Enter the STRING FAILOVER directive in the Control window command line**.

   STRING FAILOVER STRING=*100*

8. **Monitor event messages on the Global Event Display window**.

   Observe the progress of the transfer of control from the active to the backup string. During a failover, event messages are generated as processes are deactivated and as the backup processes are activated. Tables 9.5-1 and 9.5-2 identify the text of event messages associated with successful and unsuccessful string failovers and their significance.

   If the shutdown is unsuccessful, enter the STRING DELETE directive in the command line of the Control window to terminate the inactive processing as described in paragraph 9.8 and create a new string using the STRING CREATE directive.

9. **Monitor the parameter updates on the string configuration dynamic page**.

   Two key parameters indicate the successful completion of the failover. First, the RTS ID parameter will be changed to the value previously associated with the backup RTS ID parameter. Second, the backup RTS ID parameter will be set to 0. If these values are not set properly, the failover was unsuccessful. Delete the existing string and create a new string as described in step 8.

Table 9.5-1. Successful String Failover Event Message Text

Event Message Text	Significance
Shutting down __ process.	Processes are being terminated.
Successfully terminated __ Process ID __.	The process has been successfully terminated.

Table 9.5-2. Unsuccessful String Failover Event Message Text

Event Message Text	Significance
Unable to change state of __ process to __.	A problem has occurred during activation or deactivation of a process.
Unsuccessfully terminated __ Process ID __.	The shutdown was unsuccessful.

9.6 Connect to a Logical String

The STRING CONNECT directive associates a userstation with an established logical string. The following instructions explain how EOC and IST users determine which logical strings have been established at the EOC and are available for connection.

To determine which logical strings are available for connection:

1. **Open a dynamic page**.

   Click **TlmWins...** on the Control window to open the Telemetry Window Selection Box. Select a dynamic page from the list and click **OK**.

2. **Open the Data Source Selector window (see Figure 9.6-1)**.

   Position the pointer on the dynamic page, click the right mouse button and select **Data Source Selector** from the menu.

Figure 9.6-1 . Data Source Selector

The Data Source Switcher window displays all logical strings you are connected to in the Assigned Connections table. All logical strings available for connection, including strings you are connected to, are listed in the Established Connections table. Choose one of the three options from the pull-down menu above the Established Connections table to display strings with processes running on your userstation (**Local WKS**), strings with processes running on the Real-Time server (**Real-Time Server**), or all logical strings (**All**).

Logical strings with identifiers between and including 100 and 200 are shared strings available for connection. Logical strings with identifiers less than 100 represent dedicated historical replay or off-line analysis activities executing locally on the userstation. These strings are generated as a by-product of historical replay or off-line analysis and the connection to the string is accomplished by the software rather than via the STRING CONNECT directive.

The request for connection to a logical string via the CONNECT directive implies that resources on the local userstation may be employed to provide the access to the requested activity. Specify the numeric identifier for the string with which a connection is desired as well as the type of connection. There are two connection types available, mirrored and tailored. A mirrored connection inherits the configuration defined by the Ground Controller for that particular logical string. A mirrored connection is requested by specifying a mirrored configuration in the STRING CONNECT directive, such as STRING CONNECT STRING=*100* CONFIG=*MIRROR*.

A tailored connection allows you to tailor the configuration of local resources to specifications that differ from those of the Ground Controller. Upon connection, tailored users inherit the configuration of the Ground Controller. However, once the initial connection is made, the configuration of local resources may be altered and is unaffected by ground configuration changes made by the Ground Controller. A tailored connection is requested by specifying a tailored configuration in the STRING CONNECT directive, such as STRING CONNECT STRING=*100* CONFIG=*TAILOR*.

In order to assume the Ground Control privilege and modify the ground configuration for all mirrored users, establish a connection to the logical string with a mirrored configuration. If you connect to a logical string in a tailored configuration, you may not receive the Ground Control privilege in order to affect the configuration established for other users connected to the string.

To request a connection to a logical string:

1. **Open the Global Event Display window**.

   Click **Tools…** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Connect to a logical string**.

   In the Control window command line enter the STRING CONNECT directive to establish a connection to the logical string in a mirrored or tailored configuration:

   Connect to a string in a mirrored configuration:

   STRING CONNECT STRING=*100* CONFIG=*MIRROR*

or

Connect to a string in a tailored configuration:

STRING CONNECT STRING=*100* CONFIG=*TAILOR*

3. **Open a dynamic page**.

   On the Control window, click **TlmWins...** and select a dynamic page displaying user connection information.

4. **Monitor event messages on the Global Event Display window**.

   Observe the progress of the connection to the string. During string connection, event messages are generated as processes are initialized. If an error occurs during string connection, any process spawned as a result of the string connection will be terminated and event messages reflecting the shutdown of processes will be generated. If the STRING CONNECT directive is unsuccessful, investigate why and repeat step 2. Table 9.6-1 identifies the text of event messages associated with string connection and their significance:

5. **Monitor the parameter updates on the dynamic page**.

   Once the connection is established, your login name will be added to the list of users connected to the string. Your userstation will be added to either the mirrored userstation list or the tailored userstation list, depending on the configuration of your connection.

Table 9.6-1. String Connection Event Message Text

Event Message Text	Significance
Establishing __ service.	Initialization of a process.
__ process __ configured.	The string creation process has been initialized.
__ connected to String __ as __.	Successfully connected to the string in a mirrored or tailored configuration.
Shutting down __ process.	An error was encountered during the string connection. A process is being shut down.

9.7 User Authorization

There are two privileges administered by the real-time software subsystems: command authority and ground control privilege. EOC operators must acquire command authority before sending real-time commands to an EOS spacecraft, or initiating communication through the Ground Telemetry Processing Service with the NCC. Similarly, EOC operators must acquire Ground Control privilege before modifying the ground configuration of the system. In this context, ground configuration is the allocation or modification of any ground system resource within the EOC.

There is a sybase table read by the application software at request time. A process called FdUr User Roles Archive is communicated with to determine if a user and workstation are eligible to receive command authority. The FdUr User Roles Archive process will access a syase table called user_roles to make this determination. The following sections describe how the command authority and ground control privileges are managed by the system and requested by users.

9.7.1 Command Authority

In order to be eligible to receive command authority, your login ID must be in the user_roles table for the CAC user type and a particular spacecraft. The EOC System Administrator maintains the command authority user list based on operational procedures to be established by the FOT.

Userstations from which you can request and receive command authority must be listed by host name in the user_roles table for the CAC user type and a particular spacecraft. The EOC System Administrator maintains the command userstation list based on operational procedures to be established by the FOT.

To request command authority:

1. **Open the Global Event Display window**.

   Click **Tools…** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Connect to a logical string**.

   In the Control window's command line enter the STRING CONNECT directive to establish a connection to the logical string in a mirrored configuration:

   STRING CONNECT STRING=*100* CONFIG=*MIRROR*

3. **Open a dynamic page**.

   On the Control window, click **TlmWins…** and select a dynamic page displaying the current command user login identifier and command userstation identifier.

4. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify that the connection to the desired logical string was successful. If connection is not successful, investigate why by reviewing the event messages generated and repeat step 2 before proceeding.

5. **Assume command authority**.

   Enter the TAKE COMMAND directive in the command line of the Control window:

   TAKE COMMAND STRING=*100*

6. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify the progress of the authority request. If command authority is granted, an event message stating "Command Authority has changed from __ to __ for string 100" will be generated.

7. **Monitor the parameter updates on the dynamic page**.

   Observe updates to the current command user login identifier and command userstation identifier.

9.7.2 Ground Control Privilege

In order to be eligible to receive ground control privilege, your login ID must be in the user_roles table for the CAC user type and a particular spacecraft. The EOC System Administrator maintains the ground control user authorization list based on operational procedures to be established by the FOT.

Userstations where you can request and receive the ground control privilege must be listed by host in the user_roles table for the CAC user type and a particular spacecraft. The EOC System Administrator maintains the ground control userstation list based on operational procedures to be established by the FOT.

To request the ground control privilege:

1. **Open the Global Event Display window**.

   Click **Tools...** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Connect to a logical string**.

   You must connect to a string in a mirrored configuration before assuming ground control. In the Control window's command line enter:

   STRING CONNECT STRING=*100* CONFIG=*MIRROR*

3. **Open a dynamic page**.

   On the Control window, click **TlmWins...** and select a dynamic page displaying the current ground control user login identifier and ground control userstation identifier.

4. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify that the connection to the desired logical string was successful. If connection is not successful, investigate the reason why by reviewing the event messages generated and repeat step 2 before proceeding.

5. **Assume ground control privilege**.

   Enter the TAKE GROUNDCONTROL directive in the command line of the Control window:

   TAKE GROUNDCONTROL STRING=*100*

6. **Monitor event messages on the Global Event Display window**.

   Observe the progress of the request for ground control privilege. If ground control privilege is granted, an event message stating "Ground Control Authority has changed from__to__ for string 100" will be generated.

7. **Monitor the parameter updates on the dynamic page**.

   Observe parameter updates to the current ground control user login identifier and ground control userstation identifier.

9.8 Logical String Disconnection

Disconnect from a shared logical string by entering the STRING DISCONNECT directive in the Control window ECL command line and specifying the string ID from which disconnection is desired.

To disconnect from a logical string:

1. **Open the Global Event Display window**.

   Click **Tools…** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Disconnect from the string**.

   Enter the STRING DISCONNECT directive in the Control window command line:

   STRING DISCONNECT STRING=*100*

3. **Open a dynamic page**.

   On the Control window, click **TlmWins…** and select a dynamic page that includes the logical string from which you wish to disconnect.

4. **Monitor event messages on the Global Event Display window**.

   Verify the progress of the STRING DISCONNECT request. During string disconnection, event messages are generated as processes are terminated. If a process will not terminate, repeat step 2 to terminate the active process. Table 9.8-1 identifies the text of event messages associated with string disconnection and their significance.

Table 9.8-1. String Disconnection Event Message Text

Event Message Text	Significance
Shutting down __ process.	The process is being shut down.
__ terminated __ Process ID __	The process has been terminated.
User __ disconnected from String __	You were successfully disconnected from the string.

5. **Monitor the parameter updates on the dynamic page**.

   Observe the removal of your login name from the list of users connected to the string and the removal of your userstation name from either the Mirrored Userstation List or the Tailored Userstation List, depending on the connection to the string.

9.9 Logical String Deletion

Logical strings are deleted via a STRING DELETE directive specifying the string's numeric identifier, or upon completion of historical telemetry off-line analysis. Shared logical strings can only be deleted by the individual with ground control privilege for the string, after all users have disconnected from the string. In other words, to delete a shared string you connect to the string, assume ground control privilege for the string, disconnect from the string, and then delete the string as described in the following text.

For real-time, operational logical strings, the directive should specify the state of the strand to be deleted, active, backup, or inactive. If no state is entered, the active strand will be deleted by default. The active strand can only be deleted after all backup processing that exists for the string has been deleted. Backup processing is deleted by deleting the backup strand. Inactive processing is deleted by deleting the inactive strand if it wasn't successfully done during a failover.

Dedicated strings may be deleted by entering the STRING DELETE directive on the userstation where the string resides.

To delete a shared logical string:

1. **Open the Global Event Display window**.

   Click **Tools...** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Connect to a logical string**.

   You must connect to a string in a mirrored configuration before assuming ground control. In the Control window's command line enter:

   STRING CONNECT STRING=*100* CONFIG=*MIRROR*

3. **Open a dynamic page**.

   On the Control window, click **TlmWins...** and select a dynamic page displaying all available logical strings.

4. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify that the connection to the desired logical string was successful. If connection is not successful, investigate why by reviewing the event messages generated and repeat step 2 before proceeding.

5. **Assume ground control privilege**.

   Enter the TAKE GROUND CONTROL directive in the command line of the Command Control window:

   TAKE GROUNDCONTROL STRING=*100*

6. **Monitor event messages on the Global Event Display window**.

   Observe the progress of the request for ground control privilege. If ground control privilege is granted, an event message stating "Ground Control Authority has changed from__to__ for string 100" will be generated.

7. **Disconnect from the string**.

   Enter the STRING DISCONNECT directive in the Control window command line:

   STRING DISCONNECT STRING=*100*

8. **Delete the logical string**.

   Enter the STRING DELETE directive in the command line of the Control window:

   STRING DELETE STRING=*100*

9. **Monitor event messages on the Global Event Display window**.

   Verify the progress of the STRING DELETE request. During string deletion, event messages are generated as processes are terminated. If a process will not terminate, a message stating the String was unsuccessfully terminated will be generated. Repeat step 8 to terminate the active process. Table 9.9-1 identifies the text of event messages associated with string deletion and their significance.

10. **Monitor the parameter updates on the dynamic page**.

    Observe the removal of the logical string from the page.

Table 9.9-1. String Deletion Event Message Text

Event Message Text	Significance
Shutting down __ process.	The process is being shut down.
__ terminated __ Process ID __	The process has been terminated.
String __ was __	Indicates whether the string was successfully terminated.

9.10 Spacecraft Telemetry Processing

The FOS Telemetry Service receives and processes spacecraft data. All components of the Telemetry Service are initiated as part of a logical string. With respect to processing of spacecraft data, the Telemetry Service provides three distinct capabilities: data decommutation, memory dump, and state check. Decommutation is performed on both real-time and historical telemetry data. The real-time telemetry data is received from EDOS and historical telemetry data is received from the Data Management Service (DMS).

Data are received by Telemetry tasks through a set of well-known ports and addresses. The port and address used is determined by the mode; either operational, test, or training. The actual port and address is retrieved from the NameServer through the use of a service name made up of the mode and spacecraft. If the mode is operational, the data source must be real-time; if the mode is training, the data source must be simulation; and if the mode is test, the data source may be either real-time or simulation.

The memory dump function provides validation, file storage, and subsystem access to real-time spacecraft memory dump data. The spacecraft state check capability allows the FOS to store the current state of the spacecraft, monitor and compare the spacecraft's state with this baseline, and compare the spacecraft's state with its expected state.

ECL directives control the function of the Telemetry Service. Through these directives, parameters may be altered to modify data decommutation and conversion, event message generation, parameter limit processing, and spacecraft state checking.

The following sections provide detailed information about Telemetry subsystem control directives and the three functional components.

9.10.1 Directives

The following sections discuss ECL directives used to control and modify archiving, decommutation, EU conversion, limit checking, and derived parameter processing of telemetry data, and the ECL directive used to initiate spacecraft state checking. Refer to Appendix A for the syntax of ECL directives.

9.10.1.1 ARCHIVE Directive

The ARCHIVE directive controls telemetry data archiving. Modifications made by ARCHIVE directives are in effect until data archiving is redefined or the telemetry software is restarted.

9.10.1.2 DECOM Directive

The DECOM directive controls selective decommutation of telemetry parameters. The decommutation of telemetry parameters may be turned on or off for all parameters, for parameters by subsystem, or for individual parameters. Modifications made by DECOM directives remain in effect until additional modifications are made or the telemetry software is restarted.

9.10.1.3 DERIVED Directive

The DERIVED directive controls the evaluation and update of derived telemetry parameters. The DERIVED directive allows equation processing intervals to be adjusted. Setting the interval to a value of zero disables derived parameter processing. The equation processing rate is based upon spacecraft time extracted from telemetry.

Derived parameter processing involves a processing capability accommodating special computations using predefined algorithms. These simple calculations are defined in the PDB. Predefined analog, discrete, constant, or other derived values may be sources for building a new derived parameter.

9.10.1.4 DROPOUT Directive

The DROPOUT directive controls the sensing rate for detecting the loss of an input telemetry stream. The DROPOUT directive allows the dropout sensing interval to be displayed and adjusted. Setting the interval to a value of zero disables dropout detection.

9.10.1.5 EU Directive

The EU directive controls the coefficient changes of polynomial engineering unit conversion equations for telemetry parameters that have EU conversions defined in the database. The EU directive allows changes to be made to the coefficient values of EU equations and the selection of a conversion algorithm. Changes made to coefficient values for EU equations are in effect until coefficient values are redefined or the telemetry software is restarted.

9.10.1.6 LIMITS Directive

The LIMITS directive controls the limit sensing of parameters, enabling/disabling of limit messages and alarms, frequency of limit messages, and changing of individual limit values.

High/low limit checking and delta limit checking are performed on telemetry points whose parameter specification in the database specify these checks. The limits may be checked in raw or EU values.

For high or low limits checking, the actual value of the telemetry parameter is compared to the red and yellow limits defined for that parameter and reports violations in event messages. A parameter value outside yellow limits indicates that normal operational limits have been violated. A parameter value outside red limits indicates that the parameter has reached a dangerous value and that some action is required. Sets of red and yellow limits are defined in the database, but can be modified in real-time with the LIMITS directive. The selection of a set of red and yellow limits is made according to the current value of a discrete telemetry parameter called a limit switch.

With the LIMITS directive, high/low limit reporting can be turned on or off during a contact for one parameter, for a subsystem's parameters, or for all parameters.

For delta limit checking, the current value of a telemetry parameter is compared with the previous value of that parameter calculating a parameter delta (absolute difference). If the parameter delta exceeds the delta limit, a delta limit violation is reported in an event message. A delta limit can be defined in raw counts in the database and can be changed temporarily with the LIMITS directive. Changes to limits remain in effect until limits are redefined or the telemetry software is restarted.

9.10.1.7 STATE Directive

The STATE directive controls the execution of spacecraft state checking and capturing. The STATE directive allows the selection of a spacecraft table to be used for state checking; the replacement of the expected state table with current telemetry values; the start of the spacecraft state check comparison; and the switching of the telemetry channel to be state checked.

9.10.2 Data Decommutation

The Telemetry Service decommutates telemetry data received in EDU packets. Real-time data is received from EDOS via the User Datagram Protocol (UDP) over dedicated, multicast IP addresses and ports assigned per mission as defined in the applicable Operations Agreement (OA). The specific port on which each data type is received from EDOS is based on the Application ID (APID). Historical replay data is received from the DMS through a network point-to-point connection. Three types of telemetry data are received and processed by the Telemetry service: Housekeeping (HK), Health and Safety (HS), and Standby (SB).

Decommutation of telemetry data is based on parameter definitions from the database. These definitions affect the way the Telemetry Service decommutates a parameter. Several operations may be performed on a parameter after it has been decommutated. A converted value may be obtained by using one of three EU conversions: polynomial, exponential, or line segment. Either the decommutated or converted value may be high/low and/or delta limit checked using predefined limit sets. New telemetry parameters or derived parameters may be computed by performing mathematical operations on the value of one or more existing telemetry parameters. Each parameter in each EDU packet is processed according to type and the parameters sent to a Parameter Server for access by other subsystems. The EDU packet may be sent to the DMS for archiving and further analysis. Figure 9.10.2-1 is a sample display page containing decommutated telemetry values.

9.11 Memory Dump

The Telemetry Service provides the capability to receive and store spacecraft memory dump data. This data is received from EDOS via the UDP over dedicated, multicast IP addresses and ports. The Telemetry Service validates memory dump data and saves it in a file. When the dump is complete, the DMS and CMS are notified that the memory dump file is ready for storage and additional processing.

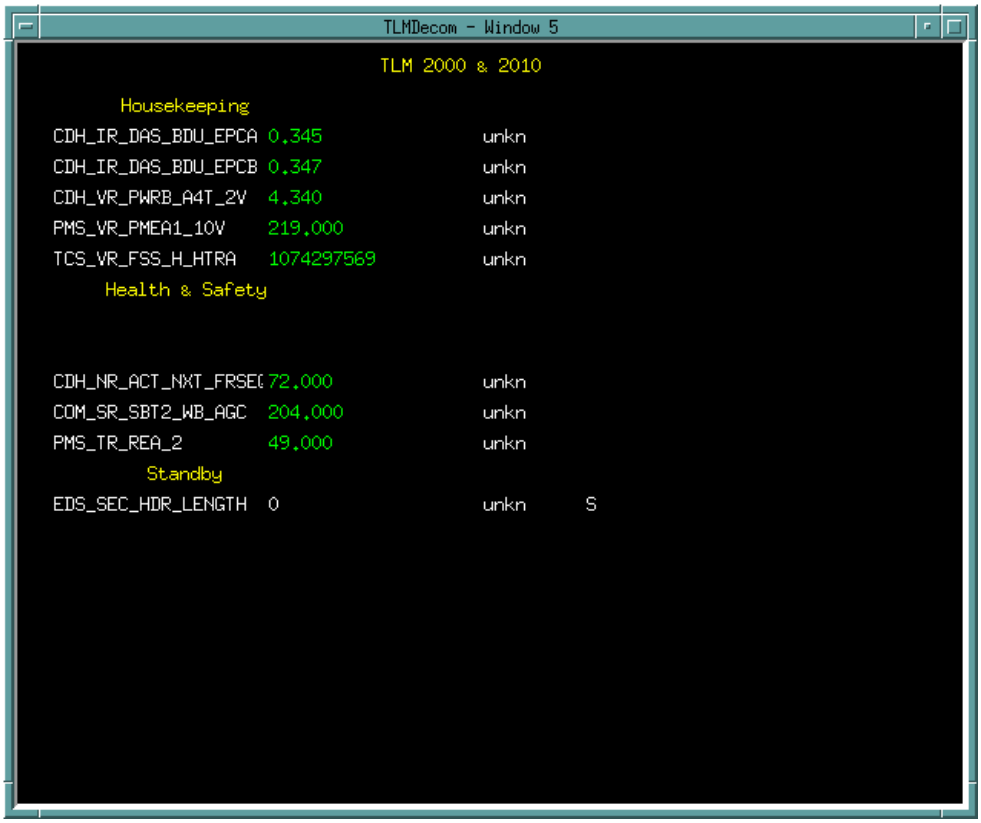

Figure 9.10.2-1. Telemetry Display Page

When a spacecraft memory dump begins, the Telemetry Service issues an event message containing the memory dump filename and expected size. Memory dump EDU packets are received by the Telemetry Service at either the 16K or 1K rate, validated according to Remote Terminal ID and word count in the EDU, and saved in the dump file. When the memory dump is complete, the Telemetry Service issues an event message containing the filename and number of words dumped. The DMS and CMS are notified that the dump is complete and the file is ready for archiving.

Three additional event messages may be issued by the Telemetry Service in the course of processing a memory dump. An event is generated if a time-out condition exists while receiving memory dump EDUs; if the memory dump is aborted before all EDUs have been received; or if the Remote Terminal ID differs from the ID expected. In any case, the event message indicates the memory dump is incomplete. After a time-out condition, the Telemetry Service is ready for another memory dump to begin.

9.12 State Check

The spacecraft State Check capability of the Telemetry Service is used to assist in back-orbit command verification. It compares the current state of the spacecraft with those expected by comparing the value of selected telemetry parameters with their expected values.

This component performs one of four functions as directed by a request received from the FOS User Interface (FUI) subsystem. First, a table of expected values for selected parameters may be loaded. These are the values to which real-time parameter values will be compared. This table of expected values will be received by the Telemetry Service from the Command Management Service (CMS) subsystem. The second function baselines selected parameters with values currently in the Parameter server. The third request may direct the telemetry subsystem to perform the state check by comparing the values of parameters in the table of expected values against those retrieved from the Parameter server. Lastly, a request may direct the Telemetry Service to change channels. This allows the Telemetry Service to retrieve parameters from a different Parameter server.

9.13 Ground Telemetry Processing

The EOC processes certain ground telemetry messages to assist in assessing spacecraft contact service sessions provided by the NCC and EDOS. Ground telemetry messages include periodic status information from the NCC and EDOS; real-time Ground Control Message Requests (GCMR) to NCC; and messages aiding EOC spacecraft clock calibration and correlation computations.

9.13.1 Directives

Directives for ground telemetry processing are entered via the command line of the Command Control window. The results of ground telemetry processing are monitored through event messages displayed on the Event Display window and parameter updates on dynamic pages. Directives applicable to ground telemetry processing are derived from the RCCONFIG, GCMR, and NCC keywords. The RCCONFIG directive has associated secondary keywords that configure the RCM subsystem. The GCMR directive has various secondary keywords that send GCMRs to the NCC. The NCC directive contains associated secondary keywords that enable and disable User Performance Data (UPD). It also has a secondary keyword specifying which NCC service is designated to conduct communication tests. The following sections describe in detail the usage of these directives.

9.13.2 Network Control Center User Performance Data

The Ground Telemetry Processing Service allows you to enable and disable UPD from the NCC UPD service via directives. It also provides a directive for conducting communication tests with the NCC UPD service. In order to send a UPD Request and conduct the communication test for the NCC UPD service, you must have proper command authority.

To send a User Performance Data request:

1. **Open the Global Event Display window**.

   Click **Tools...** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Connect to a logical string**.

   In the Control window's command line enter the STRING CONNECT directive to establish a connection to the logical string in a mirrored configuration:

   STRING CONNECT STRING=*100* CONFIG=*MIRROR*

3. **Open a dynamic page**.

   On the Control window, click **TlmWins...** and select a dynamic page which displays UPD.

4. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify that the connection to the desired logical string was successful. If connection is not successful, investigate why by reviewing the event messages generated and repeat step 3 before proceeding.

5. **Assume command authority**.

   Enter the TAKE COMMAND directive in the command line of the Control window:

   TAKE COMMAND STRING=*100*

6. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify the progress of the authority request. If command authority is granted, an event message stating "Command Authority has changed from __ to __ for string 100" will be generated.

7. **Monitor the parameter updates on the dynamic page**.

   Observe updates to the current command user login identifier and command userstation identifier.

8. **Enter the command enabling NCC UPD in the command line of the Command Control window**.

   NCC UPD *ENABLE*

9. **Monitor the Global Event Display window and dynamic page**.

   Observe event message and parameter updates to verify the UPD Request was sent to the NCC.

To conduct a communication test for NCC UPD Service:

1. **Follow steps 1 through 7 of paragraph 9.13.2, describing how to send a User Performance Data Request**.

2. **Enter the communications test command for NCC UPD in the command line of the Command Control window**:

   NCC COMMTEST *UPD*

3. **Monitor the Global Event Display window and dynamic page**.

   Observe event message and parameter updates to verify the communication test message was sent to the NCC UPD Service. Confirm that the system received the Communication Test Message echo from the NCC UPD Service.

9.13.3 Network Control Center Ground Control Message Requests

The Ground Telemetry Processing Service provides directives to send GCMRs to the NCC Reconfiguration Service, as well as a directive to select a Tracking and Data Relay Satellite (TDRS). In order to send a GCMR to the NCC Reconfiguration Service, you must have the proper command authority; to select a TDRS, you need ground control authority (refer to paragraph 9.7).

To select a TDRS:

1. **Open the Global Event Display window**.

   Click **Tools…** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Connect to a logical string**.

   In the Control window's command line enter the STRING CONNECT directive to establish a connection to the logical string in a mirrored configuration:

   STRING CONNECT STRING=*100* CONFIG=*MIRROR*

3. **Open a dynamic page**.

   On the Control window, click **TlmWins...** and select a dynamic page displaying TDRS.

4. **Monitor event messages on the Event Display Global window**.

   Observe event messages to verify that the connection to the desired logical string was successful. If connection is unsuccessful, investigate why by reviewing the event messages generated and repeat step 3 before proceeding.

5. **Assume ground control privilege**.

   Enter the TAKE GROUNDCONTROL directive in the command line of the Control window:

   TAKE GROUNDCONTROL STRING=*100*

6. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify the progress of the authority request. If the TAKE GROUNDCONTROL directive is unsuccessful, review event messages generated to determine why you did not take control and repeat step 3 before proceeding.

7. **Monitor the parameter updates on the dynamic page**.

   Observe parameter updates to the dynamic pages.

8. **Enter the RCCONFIG directive in the command line of the Command Control window**.

   RCCONFIG STRING=*100* TDRS=*TDE*

9. **Monitor events on the Global Event Display window and parameters on the dynamic pages**.

   Observe event messages to verify that the TDRS has been reconfigured. Monitor the TDRS parameter update on the dynamic page.

Ground Control Message Requests:

There are six GCMRs you can send to NCC:

1. User Reacquisition Request Message.
2. User Reconfiguration Request Message.
3. Forward Link Sweep Request Message.
4. Forward Link EIRP Reconfiguration Request Message.
5. Expanded User Frequency Uncertainty Request Message.
6. Doppler Compensation Inhibit Request Message.

The FOS provides five different User Reconfiguration Request Messages to request a reconfiguration of specified services (refer to Appendix A for additional details):

1. MA Forward Link Reconfiguration Request.
2. SSA Forward Link Reconfiguration Request.
3. MA Return Link Reconfiguration Request.

4. SSA Return Link Reconfiguration Request.
5. KSA Return Link Reconfiguration Request.

To send a GCMR:

1. **Open the Global Event Display window**.

   Click **Tools…** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Connect to a logical string**.

   In the Control window's command line enter the STRING CONNECT directive to establish a connection to the logical string in a mirrored configuration:

   STRING CONNECT STRING=*100* CONFIG=*MIRROR*

3. **Open dynamic pages**.

   Click **TlmWins…** on the Control window and select dynamic page displaying GCMR and NCC parameters.

4. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify that the connection to the desired logical string was successful. If connection is unsuccessful, investigate why by reviewing the event messages generated and repeat step 2 before proceeding.

5. **Assume command authority**.

   Enter the TAKE COMMAND directive in the command line of the Control window:

   TAKE COMMAND STRING=*100*

6. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify the progress of the authority request. If command authority is granted, an event message stating "Command Authority has changed from __ to __ for string 100" will be generated.

7. **Monitor the parameter updates on the dynamic pages**.

   Observe parameter updates to the dynamic pages.

8. **Send selected GCMRs to the NCC Reconfiguration Service**.

   Enter the following directive in the command line of the Command Control window:

   GCMR REACQUISITION LINK=*MA_FWD* SUPPORT=*FORWARD*

9. **Monitor events on the Global Event Display window and parameters on the dynamic pages**.

   Observe events on the Event Display to verify that the system received the GCMR Disposition Message and GCMR Status Message from the NCC Reconfiguration Service. Observe parameter updates on the dynamic pages.

To conduct a communication test for NCC Reconfiguration Service:

1. **Follow steps 1 through 7 of paragraph 9.13.3, illustrating how to send a GMCR**.

2. **Enter the following directive in the command line of the Command Control window.**

   NCC COMMTEST *GCMR*

3. **Monitor events on the Global Event Display window and parameters on the dynamic pages.**

   Observe events on the Event Display to verify the system received the Communication Test Message echo from the NCC Reconfiguration Service.

9.13.4 EOS Data and Operations System (EDOS) CODAs

The Ground Telemetry Processing Service provides you with information describing the operational activities of EDOS by spacecraft. EDOS sends the CODA report to the EOC every 5 seconds during a spacecraft contact session. Each report contains updated and accumulated statistics from the beginning of the spacecraft contact session to its end. Updated values can be verified on a user-defined dynamic page displaying CODAs. All CODA reports are archived in a central repository.

9.13.5 EOS Data and Operations System (EDOS) Communication Test

The Ground Telemetry Processing Service provides you with a directive for conducting communication tests with the EDOS Service. You are allowed to reconfigure the timeout value for receipt of the Command Echo Blocks from the EDOS Service.

To conduct a communication test for EDOS Service:

1. **Open the Global Event Display window**.

   Click **Tools…** on the Control window. Select the **Event Display Global** from the list of tools and click **OK**.

2. **Connect to a logical string**.

   In the Control window's command line enter the STRING CONNECT directive to establish a connection to the logical string in a mirrored configuration:

   STRING CONNECT STRING=*100* CONFIG=*MIRROR*

3. **Open a dynamic page**.

   Click **TlmWins...** on the Control window and select a dynamic page displaying EDOS parameters.

4. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify that the connection to the desired logical string was successful. If connection is unsuccessful, investigate why by reviewing the event messages generated and repeat step 2 before proceeding.

5. **Assume command authority**.

   Enter the TAKE COMMAND directive in the command line of the Control window:

   TAKE COMMAND STRING=*100*

6. **Monitor event messages on the Global Event Display window**.

   Observe event messages to verify the progress of the authority request. If command authority is granted, an event message stating "Command Authority has changed from __ to __ for string 100" will be generated.

7. **Monitor the parameter updates on the dynamic page**.

   Observe parameter updates to the dynamic pages.

8. **Enter the EDOS COMMTEST directive**.

   EDOS COMMTEST TIMEOUT=*20*

   The value 20 is an integer in seconds. The TIMEOUT subdirective is optional. If a TIMEOUT value is not specified, a default value of 8 seconds is used.

9. **Monitor events on the Global Event Display window and parameters on the dynamic page**.

   Observe events on the Event Display window to verify that a communication test message was sent to EDOS and that the system received the command echo block from EDOS.

9.14 Command Processing

The ECS Command Service provides command formatting and metering to the EDOS interface for transmission to the spacecraft. Before a command is formatted, it is validated according to the corresponding database command definition. The following items are checked as part of the validation process: command mnemonic, submnemonic(s), prerequisite telemetry values (if prerequisite checking is enabled; refer to paragraph 9.12), and criticality. In addition, before each command is sent, its binary is compared against a list of hazardous commands maintained in the database. If a hazardous command is found, processing is immediately aborted. Once a command is validated, it is formatted according to Consultative Committee for Space Data Systems (CCSDS) protocols. All commands transmitted are archived and receipt verified. Command execution

verification is then performed for commands with an execution telemetry verifier provided in the database within the database defined timeout interval. Before commanding can take place, the user must be connected to an active string and have taken command authority (see section 9.7.1). The FOP must also be initialized (see section 9.14.4).

Commands originate from real-time command directives, loads prepared by the Command Management Service and executed via the LOAD directive, and directives in EOC automated ground scripts issued from the Command Control window.

Command data blocks are transmitted to EDOS using a set of multicast ports and addresses taken from the NameServer via service names based on the operational mode and data source. If the mode is operational, the data source is limited to real-time and the Ops port and address will be used. For the test mode, the port address will be used if the data source is real-time and the training port address will be used if the data source is simulation. When the mode is training, the data source must be simulation and the training port address will be used.

Event messages are used to indicate the processing status of all commands issued to this subsystem. If a command cannot be processed without overriding submnemonic and prerequisite failures or criticality, an event message is generated and the system prompts you to allow or cancel the command. Refer to Appendix B for a list of FOS event messages.

9.14.1 Command Directives

The directives processed by the Command Service are listed and briefly described in the following text. All command directives are entered in the command line of the Command Control window. The **Allow** and **Cancel** buttons on the Command Control window are used to allow or cancel commands whose validation fails because of submnemonic, prerequisite or critical checking.

a. **CMD (or/)**. A request to format and send a real-time command from a mnemonic and optional submnemonic(s).

b. **RAW**. Request to format and send a real-time command in binary (ASCII hexadecimal) format.

c. **LOAD**. Initiate a memory load with a load identifier, partition indicator, and number of partitions.

d. **STORED**. Request for a stored command to be execution-verified.

e. **FOP**. Configure the Frame Operation Procedure (FOP) on the ground. Refer to paragraph 9.14.4 for additional details.

f. **CMDCFG**. Set command configuration parameters to select or control prerequisite checking, PLOP 1 or 2, CTIU 1 or 2, and uplink bit rate.

9.14.2 Real-Time Command Processing

Command directives sending real-time commands are entered in the Command Control window manually or received from the ground script controller via ground scripts or procedures. Command Requests at an IOT/IST also cause real time commands to be placed in the current ground script. Before a command is validated, the command source is checked to verify that the operator has current commanding authorization. The command mnemonic and submnemonic(s) are compared with the command database (CommandOdf) command definition loaded during initialization. If the submnemonic(s) had no default value or was out of range, the system prompts you to override the failure or cancel the command by clicking the **Override** or **Cancel** Command Control window button.

If prerequisite checking is enabled, up to a maximum of five command database-defined telemetry points are compared with their expected values. If all are within range, command processing continues. If one or more are out of range, an error message is displayed and the system prompts you to override the failure or cancel command processing by clicking the **Override** or **Cancel** Command Control window buttons. Prerequisite state checking can be enabled or disabled via the CMDCFG directive. The command is then checked for criticality. If the command is defined as critical in the database, you will be prompted to allow or cancel it prior to execution via the **Allow** or **Cancel** Command Control window buttons.

A final check is performed after the 1553b bus command is built against the binary hazardous command list (as defined in the database CmdFormatOdf). If the first two data words of the command match any of those in the hazardous command list, the command is immediately aborted. If not found, the command is further formatted in a CCSDS command transfer frame and metered to the EDOS interface at a user-selected rate. Receipt and execution verification is then performed for the command as described in the following sections.

9.14.3 Load Processing

Load files containing CCSDS command packets are built by the CMS and accessed via the DMS. The LOAD directive is used to initiate load processing. Loads may be divided into 4K partitions that may up uplinked at different times. The load is validated and checked for criticality. If the load contains one or more critical commands it is considered to be critical and the user is prompted to allow or cancel it via the **Allow** or **Cancel** Command Control window button.

Validation includes checking for correct spacecraft ID and that all prior partitions for this load have been uplinked. If a partition is out of order, an error message is displayed and the system prompts you to allow or cancel it by clicking the **Allow** or **Cancel** button. Prerequisite checking may also be performed against the command contained in the load that initiates the memory load onboard the spacecraft.

Receipt verification is performed for each packet in the load. When the last packet is receipt-verified, end-item telemetry verification is performed to ensure that the Cyclic Redundancy Check (CRC) for the load was successful.

ABORT may be selected to terminate the uplink of a load in progress. When a load is aborted transmission ceases immediately. It is not possible to verify what portion of the load has been uplinked beyond observing the event message display to determine how many CDBs, and hence packets, were uplinked since the load began.

9.14.4 FOP Control and Processing

The FOP provided by the Command Service permits you to keep the EOC in synchronization with the Frame Acceptance and Reporting Mechanism (FARM) on the spacecraft. The FOP and FARM together comprise the Command Operation Procedure (COP, COP1 for AM-1) and ensures and verifies that telecommand transfer frames are received by the spacecraft without omission or duplication, and in the same sequential order in which they were transmitted. For AM-1 a transfer frame contains a single packet containing a single 1553b command. For a complete discussion of COP1 and the CCSDS protocol, refer to CCSDS documentation, most noteworthy of which are:

CCSDS 202.0-B-2 Blue Book: Telecommand Part 2 Data Routing Service;

CCSDS 202.1-B-1 Blue Book: Telecommand Part 2.1 Command Operation Procedures; which includes the entire FOP state table (table 2-1), which is too lengthy to include here.

The two basic elements of COP1 are the transfer frame sequence number and the Command Link Control Word (CLCW) downlinked by the spacecraft. The spacecraft expects transfer frames to be ordered sequentially via a modulo 256 frame sequence number. FOP assigns sequential sequence numbers to each (type AD) transfer frame transmitted. The CLCW contains the spacecraft's "next expected frame sequence number" (V(R)), from which the "last accepted frame sequence number" may be computed. This is used to initially synchronize the ground with the spacecraft and to verify that frames have been received. The CLCW also contains error flags to indicate that retransmission is needed; a wait is needed (should be prevented from ever occurring by command metering); or a lockout (sequence numbers out of order to an extreme degree) has occurred.

The **FOP** directive is used to set the parameters that control FOP processing and to initialize the FOP, discussed in detail below, and to terminate processing AD commands. The FOP directive instances used to set parameters and to terminate processing are:

FOP TRANSMIT=<total transmissions count, valid range 1 to 30, default 5> Used to set the total transmission count. If set to 1 (FOP TRANSMIT=1), retransmission is disabled. A default of 5 would cause 4 retransmissions, if needed.

FOP WINDOW=<window width, valid range 1-90, default 10> Used to set the number of transfer frames that can be pending receipt verification at any given time.

FOP TIMER=<timeout value in seconds, valid range 1-60, default 30> Used to set the amount of time to wait for a CLCW after sending a command before FOP times out and either retransmits (if enabled) or issues an alarm message and reverts to the "Initial" state, after which it will require reinitialization.

FOP TERMINATE_AD Functions as a "reset" button for the FOP. It clears all of FOP's queues, including frames pending receipt verification, and reverts to the "Initial" state, after which reinitialization will be required prior to sending commands. FOP TERM (alias form) may be used any time the user wishes to "start over".

Before any commands can be sent, the FOP must be initialized and in the "Active" (ready to command) state. This implies that the ground and the spacecraft are known or assumed to be in synch with each other. This can be accomplished by synchronizing the ground with the spacecraft (the norm), or the spacecraft with the ground. The FOP directive cases used to initialize the FOP are:

1. **FOP INIT CHECK**. This synchronizes the ground to the spacecraft by receiving a CLCW and using its V(R) for the sequence number of the next frame to be sent. If a nominal CLCW (no error flags set) is not received before the selected timer expires, initialization will be considered unsuccessful. Should this happen, it is possible that CLCWs are not available, requiring "blind commanding", discussed in item 3 below.

2. **FOP INIT V(R)=<*next expected frame sequence number*>**. This synchronizes the spacecraft to the ground by sending a type BC control command (set V(R)) to set its V(R) value to the requested value. Control commands are not subjected to the frame sequence number check performed for regular (type AD) commands. After the set V(R) command is sent, a nominal CLCW is expected with a V(R) value identical to the one entered in the directive.

3. **FOP INIT NOCHECK**. Do not wait for a CLCW, start commanding with whatever FOP's next frame sequence number to be sent is set to. This would usually be done when telemetry, and hence CLCWs, are unavailable ("blind commanding"). A typical scenario would be to disable retransmission (**FOP TRANSMIT=1**), set the time out to one second (**FOP TIMER=1**), enter **FOP VS=<n>** directive to tell FOP what sequence number to use next, then **FOP INIT V(R)=<n>** to send a set V(R) command (knowing that it will not be acknowledged by a CLCW but will time out and hence finish quickly), and finally, **FOP INIT NOCHECK**. The queue size and the time out should then be set to the maximum allowed values to allow as many commands to be sent as possible before a time out occurs and the FOP goes into retransmission mode. Then, the above scenario would have to be repeated. FOP TERM could be used instead of waiting for FOP to time out, allowing the user to leave the time out parameter at its nominal value.

4. **FOP INIT UNLOCK**. This would only be used when the CLCW lockout flag is set. It sends a BC Unlock command to the spacecraft and then waits for a nominal CLCW, leaving the FOP in the "Active" state.

The FOP is always in one of six states, as indicated on the Command Parameters dynamic page (see Figure 9.14.4-1). They are:

1. **Initial**. FOP has not been initialized, commanding not permitted. This would be expected after FOS has been brought up but no FOP directives entered.

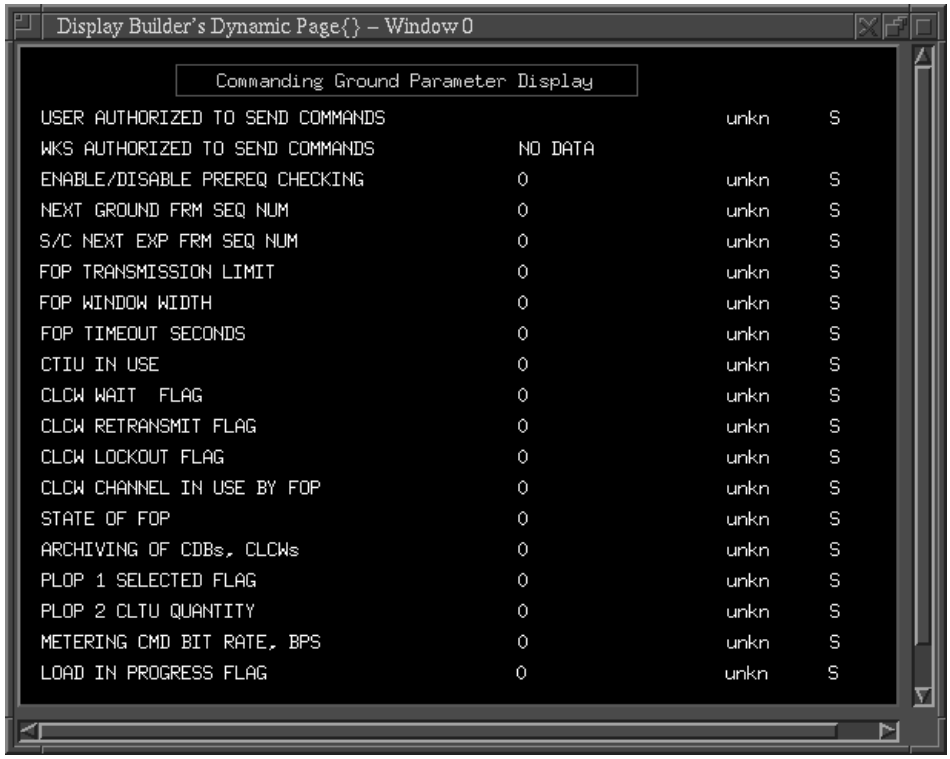

Figure 9.14.4-1. Command Parameters Display

2. **InitializingWithBCFrame**. Type BC frame sent (unlock or set V(R)) but nominal CLCW not yet received.
3. **InitializingWithoutBCFrame**. FOP INIT CHECK entered and waiting for nominal CLCW.
4. **RetransmitWithWait**. A CLCW has been received with the retransmit and wait flags set. The FOP is in retransmission mode and no new commands are permitted.
5. **RetransmitWithoutWait**. A CLCW has been received with the retransmit flag set. The FOP is in retransmission mode and no new commands are permitted.
6. **Active**. FOP is initialized and ready to send commands or commands have been sent and are pending verification but no problems have occurred.

As FOP sends frames, it saves them in a queue awaiting CLCW verification. The size of the queue is referred to as the "sliding window width" and is adjustable via the FOP directive (FOP WINDOW=<window width>mentioned above). As CLCWs arrive, an incrementing V(R) indicates frames verified and removes them from the queue. If a frame is received by the spacecraft with an out of order sequence number, it is rejected and the "retransmit flag" bit is set in the CLCW. This causes FOP to retransmit all the frames in its queue and disable transmission of new frames. The number of transmissions is also controlled by the FOP directive (FOP

TRANSMIT=<transmissionLimit>). It sets the total number of transmissions allowed, so setting it to one disables retransmission. In order to keep from declaring that the link has failed when it is in fact getting frames into the spacecraft, the transmission count limit applies only to the first (oldest) frame in the queue. Once that frame is acknowledged, the count is reset, even though the remaining frames on the queue have been transmitted, possibly more than once. The effect is that the transmission count can be considered to be associated with the queue, rather than with each frame.

FOP also expects to get CLCWs periodically and if it does not receive one within a time limit selected by the user via the FOP directive (FOP TIMER=<timeoutValueInSeconds>), it will begin retransmission, if enabled by the user and if frames are in the verification queue. The timeout value is also used while initializing the FOP and waiting for CLCWs, as previously discussed.

The spacecraft has two command and telemetry interface units (CTIU) to which commands may be directed. Each CTIU has its own CLCW, referred to as "I" and "Q" channel. Normally the Q channel CLCW is associated with CTIU-1 but is changeable via commands to the spacecraft and the ground system configuration at EDOS and WSGT. The destination CTIU and the CLCW channel to be used for frame acceptance verification are selected via the CMDCFG directive. It is up to the user to know which channel is associated with which CTIU and which CTIU is primary (active). If a switch is made from one CTIU to another, it is advisable or necessary to switch the CLCW channel that will be used by FOP (CMDCFG CLCW=I|Q) and then to reinitialize the FOP. It is also highly advisable to make such switches, when possible, when no commands are pending receipt verification.

9.14.5 PLOP1/PLOP2

The Physical Layer Operation Procedure (PLOP) is provided by the Command Service. The CMDCFG directive is used to select PLOP1 or PLOP2. For PLOP1 a Command Data Block (CDB) to be sent to EDOS always contains a single Command Link Transmission Unit (CLTU) followed by an 8 byte gap sequence of all zeroes.

PLOP2 for real-time commands is identical to PLOP1 except that there is no gap at the end.

When PLOP2 is selected and a load is being uplinked, the number of CLTUs to be placed in a CDB is determined by the CLTU quantity selected when PLOP2 was selected. For example, if the CLTU quantity selected was 30, 30 CLTUs would be accumulated, placed in a CDB, and sent to EDOS. The detection of the end of the load before 30 was reached would cause the CDB to go out immediately, regardless of the number of CLTUs placed in the CDB thus far.

WARNING

If the FOP sliding window size is set to less than the PLOP2 CLTU quantity when a load is being uplinked in PLOP2, the Command Service (unless the load is small) could be hung! This is because the FOP queue will be full with FOP waiting for verification status on frames sent and will not forward any more CLTUs to the PLOP portion of the Command Service. PLOP will be waiting for more CLTUs to place in a CDB because it has not yet accumulated

the desired amount or detected the end of the load. If this occurs, you will have to wait for FOP to time out and "retransmit" the frames (even though they have not yet been transmitted). This will result in frame sequence errors, etc., and the FOP will probably have to be reinitialized.

9.14.6 Command Failover Processing

When Command Service software is initialized it checks to see if it is "active" or "backup." If active, it checkpoints its configuration state (parameters that may be set via the CMDCFG or FOP directives) and thereafter whenever the state is changed. If backup, it retrieves the active process configuration state and then receives all directives in parallel with the active service. If and when the backup becomes active, it will be in synchronous regarding configuration state and probably synchronous regarding dynamic state (verification queues, etc.). You are urged to bring up a backup string before any commanding is done on the active string to ensure that the two strings will be synchronous should a failover be necessary.

9.14.7 Execution Verification

When commands are defined in the command database, a telemetry parameter may be associated with the command and a value it is expected to take as a result of the command being executed. The length of time allowed for verification (time out value in seconds) is also defined in the command database. When a command is sent, it is added to a queue of commands awaiting execution verification that commences as soon as the command is metered to EDOS. The queue is checked by a timer set for the minimum execution verification timeout of the queue. An event message for each command will be generated indicating a successful verification or timeout, at which time it is removed from the queue.

This page intentionally left blank.

10. Off-Line Services

Figure 10-1 illustrates the flow of off-line and real-time analysis data through the FOS.

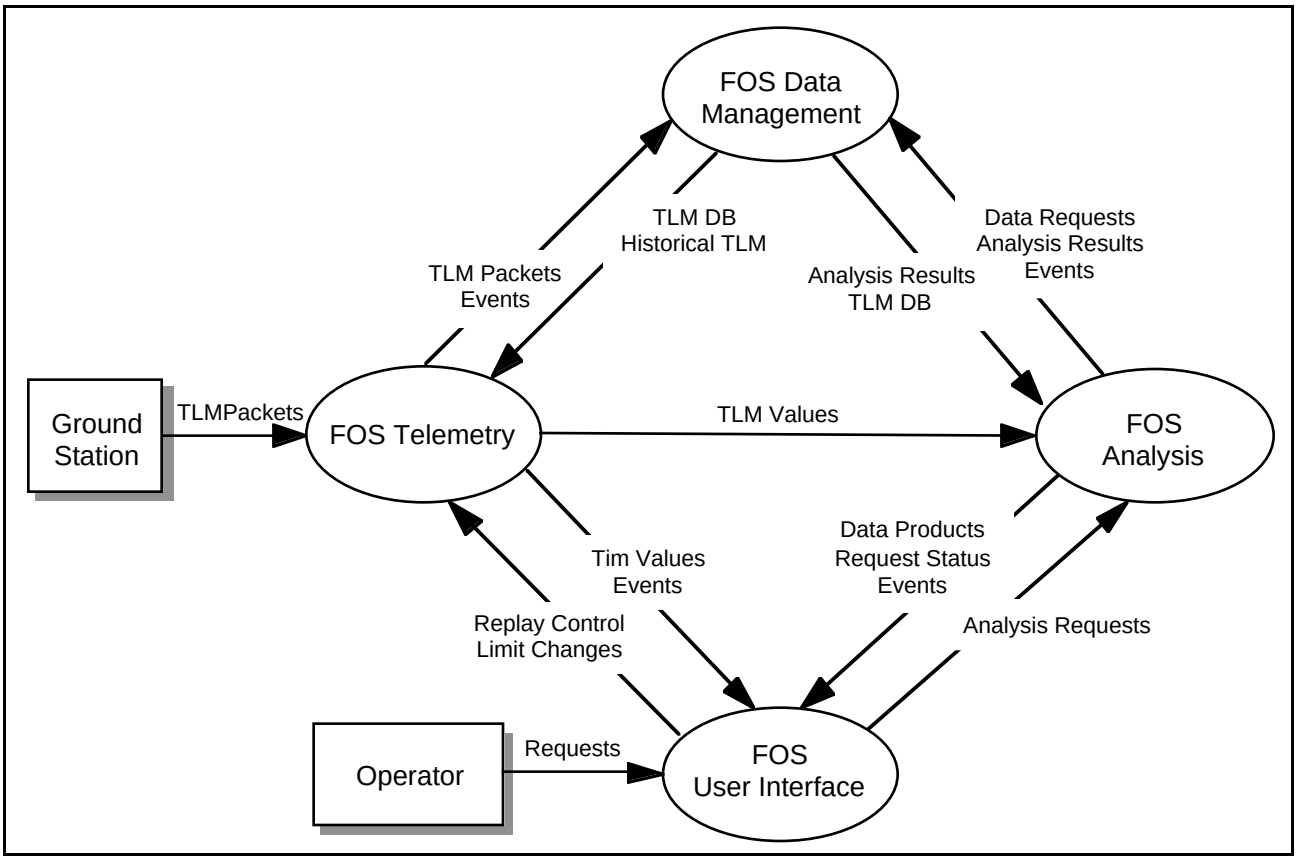

Figure 10-1. Analysis Data Flow Diagram

10.1 Real Time Analysis

10.1.1 Clock Correlation

The Clock Correlation process is responsible for evaluating the bias between the time kept by the onboard spacecraft clock and the UTC time kept on the ground system. This bias, or clock error, is evaluated by two separate algorithms: the Return Data Delay (RDD) and the User Spacecraft Clock Calibration System (USCCS).

A clock correlation configuration file containing various time parameters is maintained for the calibration process. This file must be initialized prior to performing the first clock correlation. Once configured, these time parameters will be used in all calibrations of the clock correlation

process. The user needs only to reconfigure this file if there are changes in the time parameters. To initialize the clock correlation configuration file, execute the clock correlation ODF editor, FaCcClockConstantEditor. A menu appears for the file configuration (see Figure 10.1.1-1). Select the parameter and enter the corresponding value, in Julian time. Once the file is configured, the clock correlation process can be performed.

Constants	Default Time Values	Descriptions
1) PropogationDiff:	: 0	The Difference between a PN epoch's forward and return propogation delays.
2) XpndrDelayDiff:	: 0	The Difference between a PN epoch's forward and return Transponder delays.
3) DelTcvcdu_packet:	: 0	The time delay associated with demodulating CVCDU into an EDU.
4) DelTwindow:	: 0	The time delay between the packet at the Spacecraft and the opening of epoch sampling window.
5) DelTtgt_to_EDOS:	: 0	The delay from the time when the return telemetry's electronic signal leaves the TGT low-rate black switch until it is time-tagged at EDOS.
6) TDRSrtn_tlm_delay:	: 0	The time delay experienced by return telemetry as it passes through TDRS.
7) Tuser_RDD:	: 0	The amount of time required to build a spacecraft packet.
8) Tuser:	: 0	The time required for the Time Transfer Epoch to induce the CTIU to read the spacecraft clock.
9) Tepoch_cycle:	: 23	The nominal time delay from the departure of a fwd PN epoch until the arrival of a rtn PN epoch.
10) Tepoch_period:	: 0	The nominal time period between successive fwd PN epochs.
11) SCfwd_xpnder_delay:	: 0	The time delay from the arrival of a fwd epoch at the Spacecraft antenna until the time transfer epoch is emitted from the transponder.
12) SCrtn_xpndr_delay:	: 0	The time delay from the emission of a time transfer epoch until the return epoch signal reaches the spacecraft transmitting antenna.
13) NominalClockFrequency:	: 0	The nominal value of the Master Oscillator frequency.
14) AdjustmentDeterminant:	: 0	The threshold value of the difference between the expected clock bias and the actual clock bias. If this threshold value is exceeded, then it is assumed an adjusment has been made since the last measurement.
15) ErrorThreshold:	: 0	The positive upper limit on the clock error. When the clock error has reached or surpassed this value, a clock adjustment is needed.
16) USCCSErrorAccuracy:	: 0	The accuracy of any given USCCS clock bias calculation. USCCS is said to provide down to 4 microsecond accuracy in its calculations.
17) RDDErrorAccuracy:	: 0	The accuracy of any given USCCS clock bias calculation.
18) SpeedOfLight:	: 0	The speed of the electronic signal in units of KM/day. Time Value should be set to 3e+06.
19) AdjustCmdMneumonic:	: none	The Command Mneumonic for tweaking the spacecraft clock by adjusting the Master Oscillator's frequency value.
20) AdjustSubMneumonic:	: none	The Command Sub-mneumonic for XCcAdjustCmdMneumonic.
21) AverageIChannelRCTD:	: 0	The average value for the previous pass of the time delay experienced by the return I channel telemetry signal as it passes through the TGT to the last output connector that leads to the multiplexer/demultiplexer(MDM) at the TGT Data Interface System(DIS).
22) AverageQChannelRCTD:	: 0	The average value for the previous pass of the time delay experienced by the return Q channel telemetry signal as it passes through the TGT to the last output connector that leads to the multiplexer/demultiplexer(MDM) at the TGT Data Interface System(DIS).
23) LatestAdjustmentTime:	: 0	The estimated absolute time of the most recent frequency adjustment.
24) InitialError:	: 0	The estimated clock error at the time of the most recent frequency adjustment.
25) InitialFrequency:	: 0	The estimated clock frequency at the time of the most recent frequency adjustment.
26) MoAgingRate:	: 0	The current value of the MO aging rate, i.e. the rate of change of the clock frequency over time.

27) CommandRequestFlag:	: 0	Flag used to determine whether or not the Command Request to adjust the clock is automatically generated by this process. If the flag is true, and the clock needs adjusting, the Command Request will be generated. Default Time Value is enabled.
28) TableLoadFlag:	: 0	Flag used to determine whether or not the Command Request to adjust the clock is automatically generated by this process. If the flag is true, and the clock needs adjusting, the Command Request will be generated. Default Time Value is enabled.
29) FaXCcPreviousErrorValues:	: Error Sample Menu	The vector of previous clock error calculations; used in regression to find other process values.

Figure 10.1.1-1. Clock Correlation Constant Editor Menu

The clock correlation process must be executed for each contact in which a calibration is desired. Upon startup, the process connects to a real-time string. Both the real-time RDD and the post contact USCCS processes are started with an ECL directive with the following syntax:

CC START STRING = <real time string id, e.g. 100>

CC START SCID = <spacecraft id, e.g. AM1> MODE = <ops, training, or test>

The clock correlation process will execute and a clock bias will be calculated. Monitor the Events Display for status on the clock correlation process. Contacts containing ranging data, used for the clock bias calculation, will generate OPM-66 messages (see The Clock Correlation Process, section f, below). These messages will automatically terminate the clock correlation process. For contacts without ranging data, the user must manually terminate the process by typing the ECL directive with the following syntax:

CLOCK END

The Clock Correlation Process

1. **Clock Controlling**. Rather than using clock jumps, the spacecraft clock will be steered by adjustments to the Master Oscillator frequency. This change in frequency adjustment will occur when the clock bias has exceeded some predetermined value, set in the clock configuration file.

2. **Return Data Delay (RDD)**. The RDD method calculates the clock bias by using the TDRS Ground Terminal (TGT) time stamps on the telemetry packets. Various time delays, set in the clock configuration file, are subtracted from the Ground Receipt Time to determine an estimated telemetry packet time. This estimated packet time is compared to the actual Spacecraft Time, the difference between the two being the RDD clock bias.

3. **User Spacecraft Clock Calibration System (USCCS)**. The USCCS method makes use of Pseudo-Noise (PN) Ranging Epochs generated by a TDRS Tracking Service. These epochs are transmitted from the TGT at a rate of one every 85 milliseconds (+/- 1 msec). The rate varies +/- 1 msec depending on the forward link frequency and Doppler compensation. Once a forward epoch is received by the spacecraft, the transponder sends a Time Transfer Epoch to the CTIU, which in turn places a spacecraft clock reading into the

telemetry stream. Once the Time Transfer Epoch is generated, the transponder sends a return PN epoch to the TGT.

4. **Implementation**. Due to use of predicted range values in the real-time calculation, the RDD method is guaranteed to generate clock bias measurements accurate to only 20 milliseconds. During the initial phase of a spacecraft mission (e.g., during orbit acquisition), this will be a sufficient accuracy specification to provide information about the spacecraft clock. This will allow the FOT to steer the clock to 20 millisecond accuracy. Once true spacecraft time is known to be within +/- 40 msec, the USCCS algorithm can be used. In nominal operations during a real-time pass, the RDD method will announce a calculated clock bias for each passing minute, via an event message. Once the contact is over, the process will determine an average RDD calculated bias for the entire pass. This average value is reported to the FOT through an ASCII report and an event message. The data is available for analysis through the Report Browser, see section 7.13. The report has the format X1_RDD_X2.txt, where X1 refers to the spacecraft ID and X2 is the time stamp. If a spacecraft's mission requirements for onboard clock accuracy are finer than the 20 millisecond inaccuracy of the RDD method during nominal operations (e.g., AM1's 100 microsecond accuracy is finer than the 20 millisecond inaccuracy of the FOS configuration), the clock bias will be truncated by the inherent RDD inaccuracy (currently 20 msec), and thus will contain a zero-value. These minute-by-minute calculations are to be used as "sanity checks" for nominal operations.

5. **Return Channel Time Delay (RCTD)**. RCTDs are stored in the operational data base under the table name fos_opm62. The table contains time delay measurements describing the amount of time required for the telemetry signal to travel from the TGT antenna to the output of the Low Rate Switch. The values are used as input for calculating clock correlation during real-time contacts using the RDD method. The fos_opm62 table is configurable and can be updated by the Data Base Administrator (DBA).

6. **Time Transfer Message**. The TGT forward and return ranging epochs are time-stamped and recorded at a FOS required rate of one per second. The recorded epochs are sent to the NCC, which in turn forwards them to the EOC as a Time Transfer Message (TTM), also known as OPM-66. This message also contains the forward and return Range Zero Set (RZS) time delays. These delays describe the amount of time the TGT forward and return epoch signals spend within TGT. The measurement is from the Modulator to the RZS translator in the antenna and back to the Integrated Receiver for a given tracking service. In addition, the TTM can provide up to 4.5 minutes of data to the USCCS process. If more time transfer data is required, up to five independent, non-over lapping, two way tracking services may be scheduled in a single contact. The Clock Correlation process receives the TTM at the end of the tracking service. Once the process receives the recorded epochs, it reconstructs the unrecorded epochs. See reference 531-TR-001, User Spacecraft Clock Calibration System, for information on methods used to reconstruct unrecorded epochs. The process then begins matching forward and return epochs to the telemetry packet containing the spacecraft clock reading associated with that epoch pair. A USCCS clock bias will be calculated for each telemetry packet containing a spacecraft clock reading. The process generates an average of all the USCCS calculations, and reports them

to the EOC via an ASCII report and event message. The report can be viewed via the Report Browser, see Section 7.13.

7. **Automated Clock Correlation**. In the event the USCCS-calculated bias exceeds some predetermined threshold, set in the configuration file, the Clock Correlation process will determine that a frequency adjustment is required. By default, the process will generate one Command Request to adjust the MO frequency and another Command Request to update the TDRS On-board Navigation System (TONS) MO frequency parameter. These products are then posted for approval/disapproval by the Command Activity Controller. This automated functionality can be enabled or disabled by an ECL directive, using the following syntax:

   CC ADJUST ENABLE (or DISABLE)

   CCTABLE ENABLE (or DISABLE)

10.1.2 Solid State Recorder Manager

The Solid State Recorder (SSR) Manager will supply Command Requests to the Command Activity Controller to efficiently playback and replay science data from the Solid State Recorder Data Memory Unit (DMU) buffers. Housekeeping data playback from the DMUs is not managed by the Solid State Recorder Manager. The SSR Manager will also provide displays of the status of the SSR buffers during the contact. This status will include problem and solution messages, as well as missed data locations. Refer to Section 9 for additional information on employing real-time services to monitor and control historical replay activities for a given spacecraft.

The SSR DMU buffers are dumped during real-time TDRS contacts. Command requests are generated to playback the data based on the time available during the support and the amount of data in each buffer. If all the data cannot be dumped during the contact, the SSR Manager will determine what percentage and/or order of each buffer to be played back. See configuration files for a further description on how the buffers are dumped. Based on this information, the SSR Manager will issue command requests to playback the data. Theses requests are issued 5 minutes prior to the upcoming contact.

After the contact, the SSR Manager sends the actual buffer counters to Planning and Scheduling (PAS) for timeline updating. The actual counters are compared against the predicted values. If any data is missing, the SSR Manager will replay the buffer(s) at the next available opportunity. Playback reports are also generated and can be viewed through the Report Browser. See Section 7.13 for use of the Report Browser.

To start the Solid State Recorder Manager:

1. From the Control Window command line enter the following ECL directive:

   STRING CONNECT STRING=<real time string id> CONFIG=MIRROR

2. After the string is successfully connected, enter the ECL directive:

   EA ENABLE STRING=<real time string id> (EA is an alias for the ECL directive, Expert Advisor)

3. The SSR Manager should be up and running at all times on the designated SSR machine, "giraffe". Verify this is true. If not, following the string enable, start the SSR Manager from "giraffe". The script is located in the "scripts/setup" directory, and is executed by typing the following command: **SSRStartUp**. When executed, the script initializes the FaIeSSR for interaction between the RTworks Inference Engine and FOS processes. If the real-time string is down, disable and enable the string.

4. Click **Tools** from the Control Window.

5. Select **SSR_Analysis** to open the SSR Manager Top display (see Figure 10.1.2-1).

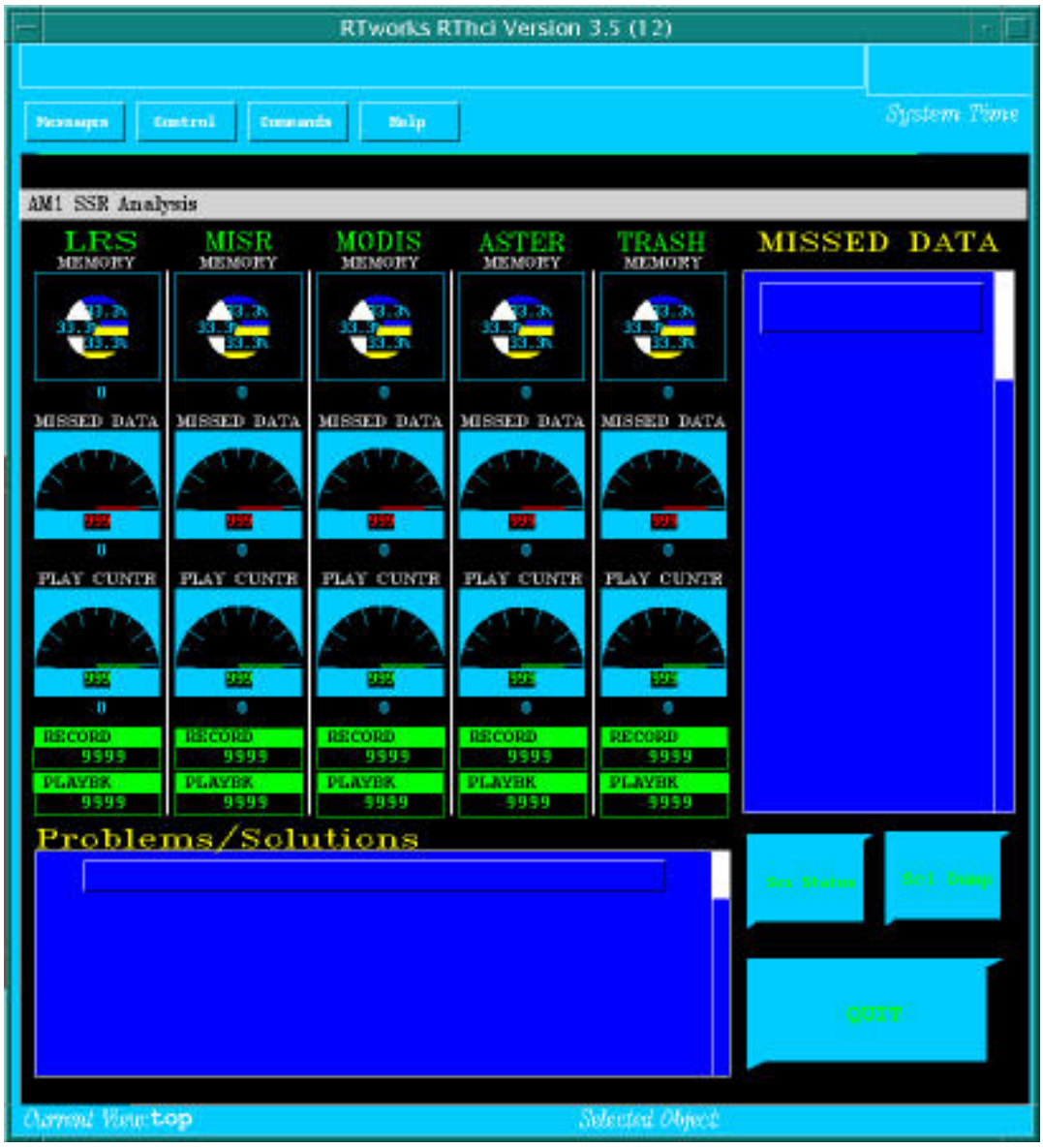

Figure 10.1.2-1. SRR Manager Top Display Window

6. Click on the dials or graph in the individual buffer row to bring up the individual buffer window. For example, if the MODIS Buffer is selected, the MODIS Buffer Schematic Page opens (see Figure 10.1.2-2). The rest of the individual buffer displays (LRS, MISR, ASTER, and Trash) have the same format as the MODIS display.

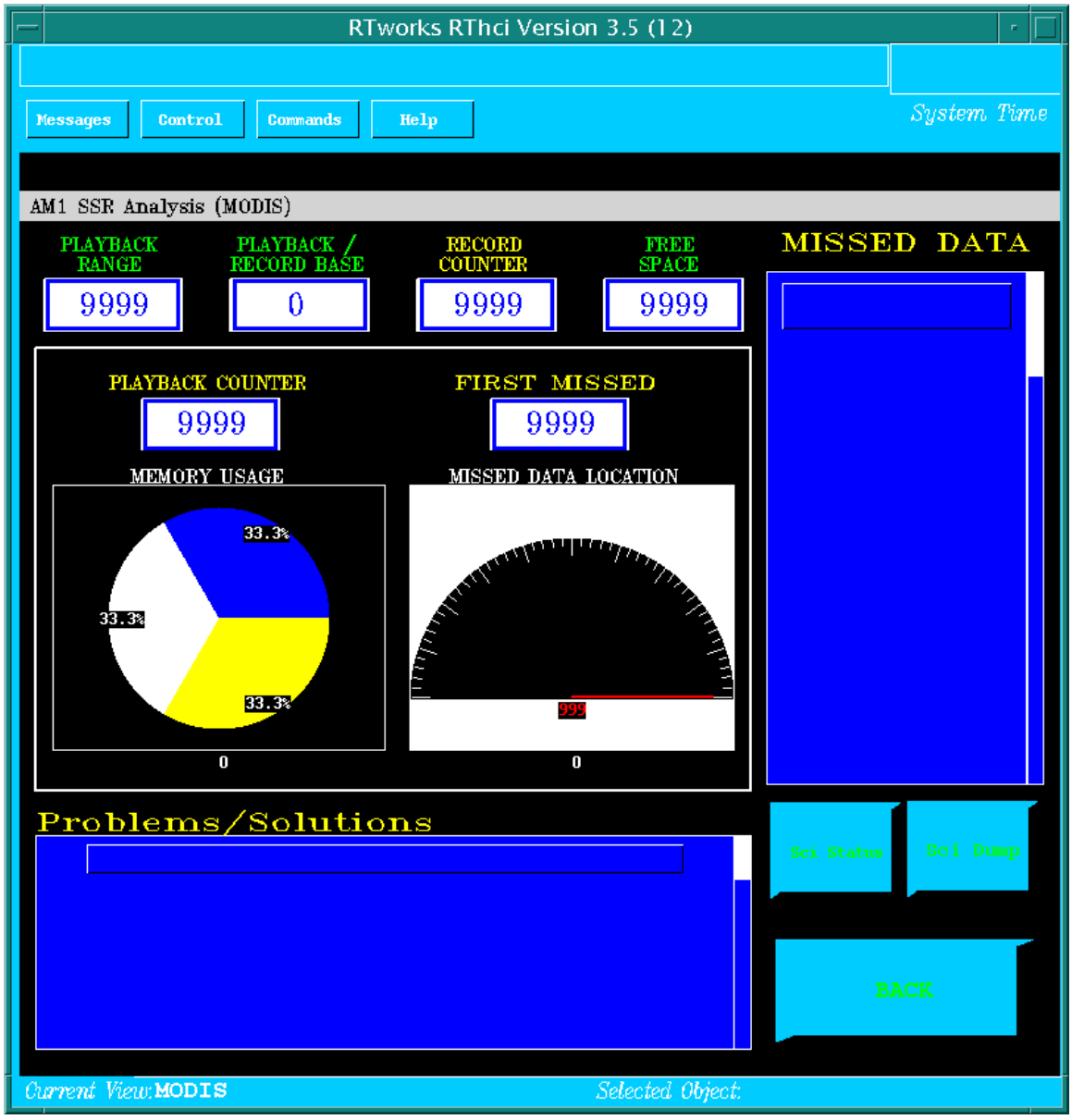

Figure 10.1.2-2. MODIS Buffer Display Window

The "Problems/Solutions" box and the "Missed Data" box on the individual buffer pages contain the same information as the boxes on the SSR Manager Top display page. The **BACK** button brings you back to the SSR Manager Top display window. The **Sci Status** button brings up the

AM1 SSR Health and Safety window (see Figure 10.1.2-3). The **Sci Dump** button brings up the AM1 SSR Science Dump window. Development of this display is TBD.

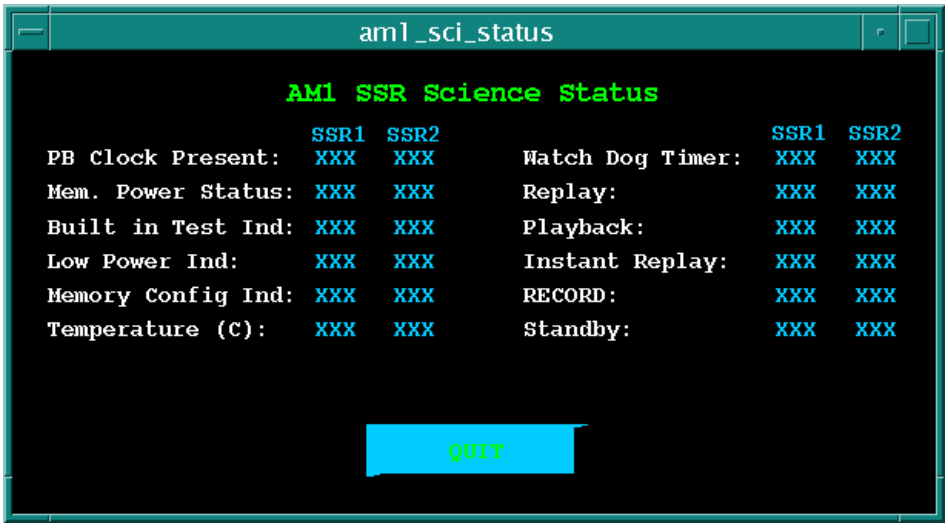

Figure 10.1.2-3. AM-1 SSR Science Status Window

Configuration Files. The configuration files described in the following text are used by the SSR Manager to build the playback and replay command requests. The objective is to replay and playback as much science data as possible during the contact according to the following configuration files as defined by the FOT. The SSR Manager will attempt to dump all the data first. If this is impossible, the SSR Manager will use the percentage file, then if unsuccessful, the buffer order file. In the event data is missing during a playback, the threshold file is used to determine how the data is to be replayed. All configuration files are located in the directory: /net/giraffes/data/rtworks_SSR

1. **The percentage.cfg file**:

   The percentage.cfg file is used if the total amount of science data stored in all the buffers cannot be played back. Percentages for each buffer can be set. The SSR Manager will then try to playback the stated percentage of the stored science data of each buffer using these values.

   The file is an ASCII file containing five numbers delimited by a space. The order of the buffer percentages is: LRS, Misr, Modis, Aster, and Trash.

   For example:

   .95 .75 .75 .60 1.0

   95% of LRS will be played back.

   75% of Misr and Modis.

   60% of Aster.

100% of Trash.

2. **The bufferOrder.cfg file**:

   The bufferOrder.cfg file is used if all the science data stored in the buffers cannot be played back or the percentage of science data for each buffer cannot be played back during the upcoming pass. The bufferOrder.cfg file sets the order of the buffers. The SSR Manager will use this order to determine which buffers will be dumped. The SSR Manager walks through this buffer list. If there is enough time to playback the whole buffer, the SSR Manager will build the appropriate commands. The SSR Manager will try and dump as many buffers as time allows in the contact.

   This file is an ASCII file containing five numbers delimited by a space. These numbers are the buffers' Buffer ID. The Buffer ID's are: Trash0, LRS1, Misr2, Modi3, and Aster4.

   For Example:

   3 2 1 4 0

   The order will be Modis, Misr, LRS, Aster, and Trash.

3. **The thresholds.cfg file**:

   The thresholds.cfg file sets the replay percentage thresholds for each buffer. The percentage represents the ratio of the amount of data to be replayed to the total amount of previously played back data currently in the buffer. Once this threshold has been reached, one replay will be issued for the total amount of previously played back data in the buffer, instead of individual replays for each area missed. The file is an ASCII file containing five numbers delimited by a space. The order of the buffer threshold percentages is: LRS, Misr, Modis, Aster, and Trash.

   For Example:

   .40 .65 .65 .75 .30

   If 40% of the LRS buffer needs to be replayed, replay all of the previously played back data.

   65% of Misr and Modis.

   75% of Aster.

   30% of the Trash buffer.

10.1.3 The Decision Support System (DSS)

The Decision Support System (DSS) will monitor the health and safety of the spacecraft through telemetry and ground support data. Data is received through parameters servers. Through data analysis, the DSS will determine the current state of the spacecraft and track spacecraft state changes. The system will detect and isolate anomalies, and provide recovery recommendation to the user. It will also generate event messages and logs as part of its output to the user. Data is received through the parameter servers.

To start the DSS:

1. Telnet to "giraffe".
2. Set display back to current userstation or xterm.

   Source the Rtworks setup file:

   /net/giraffe/data/rtworks35/bin/rtinit.csh

3. Change to DSS home directory: cd /data/mcs_2.01/apps/ie.
4. Start RTworks inference engine process in the background: **rtie &**.

   The RTie interface comes up and will already be in run mode (it takes a couple of minutes to completely initialize). **Note:** Steps 1 and 3 will change if RTworks resides on a work station other than "giraffe".

   a. **RTworks Inference Engine (RTie) Window.** This window displays state transition related to rules firing (see Figure 10.1.3-1).

   b. Review messages to analyze the health and safety of the spacecraft.

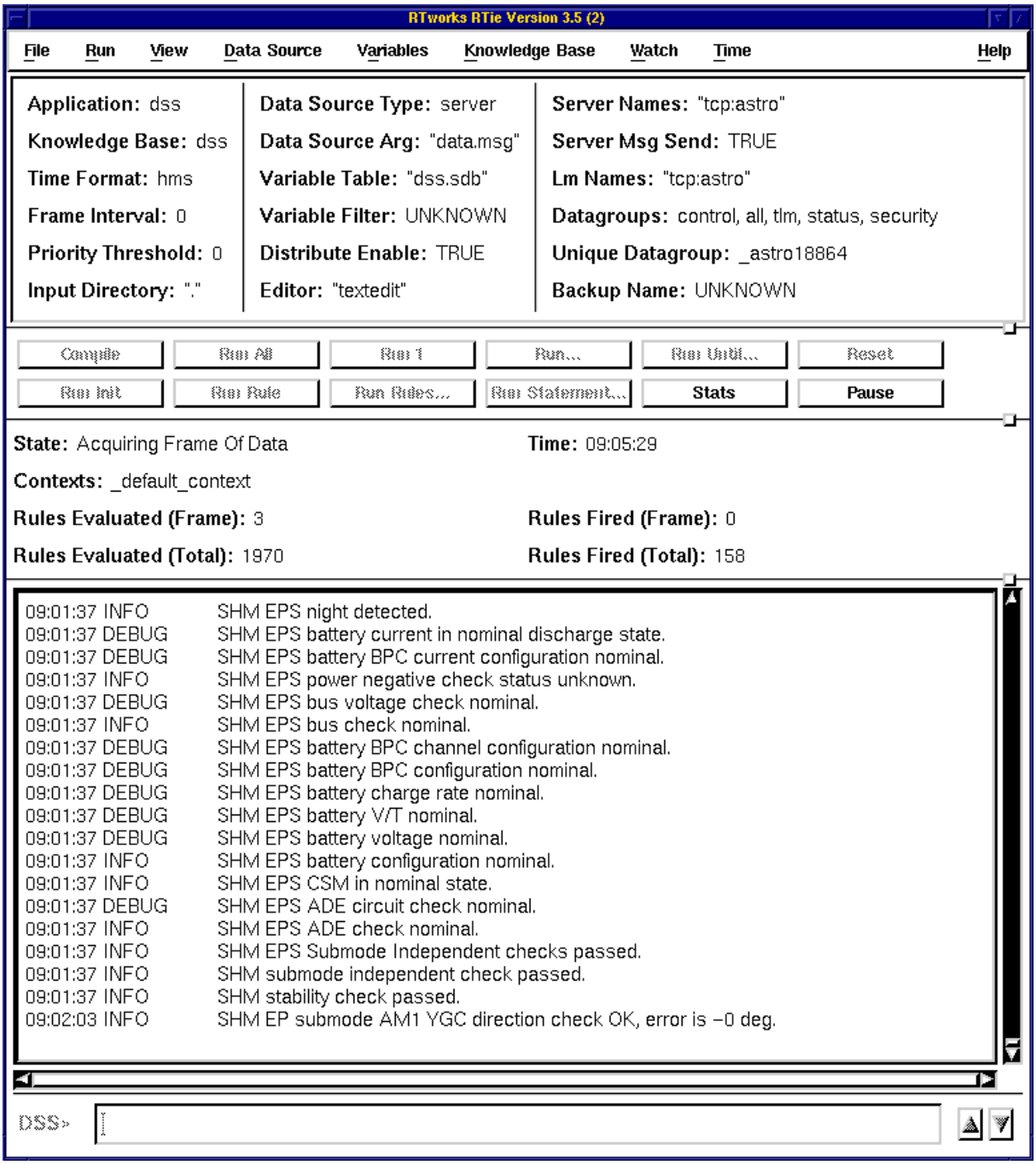

Figure 10.1.3-1 RTie Window

10.2 Off-Line Analysis

The Analysis Request allows you to create a dataset of telemetry covering time spans of valid archived housekeeping data. The request allows you to sample telemetry by every N samples, where N is a number from 1 to 32767, or by change only. You can plot data contained with a dataset and create reports containing plots and tables.

User Selectable Statistics are minimum-maximum-mean standard deviation statistics calculated over an interval between 1 second and 24 hours. You may select interval lengths for user statistics while building the analysis request. The resulting data is placed in a dataset that may be formatted as a graph or table. The Queue Manager and event messages provide status of the request. Each analysis request will result in the creation of one or more logical strings. Users should not shutdown user stations with active logical strings associated with analysis requests. Refer to Section 9.2 for more information on logical strings.

10.2.1 Build an Analysis Request

To build an analysis request:

1. Click **Tools** from the Control Window.
2. Select **Analysis_Request_Builder** to open the Analysis Request display (see Figure 10.2.1-1).

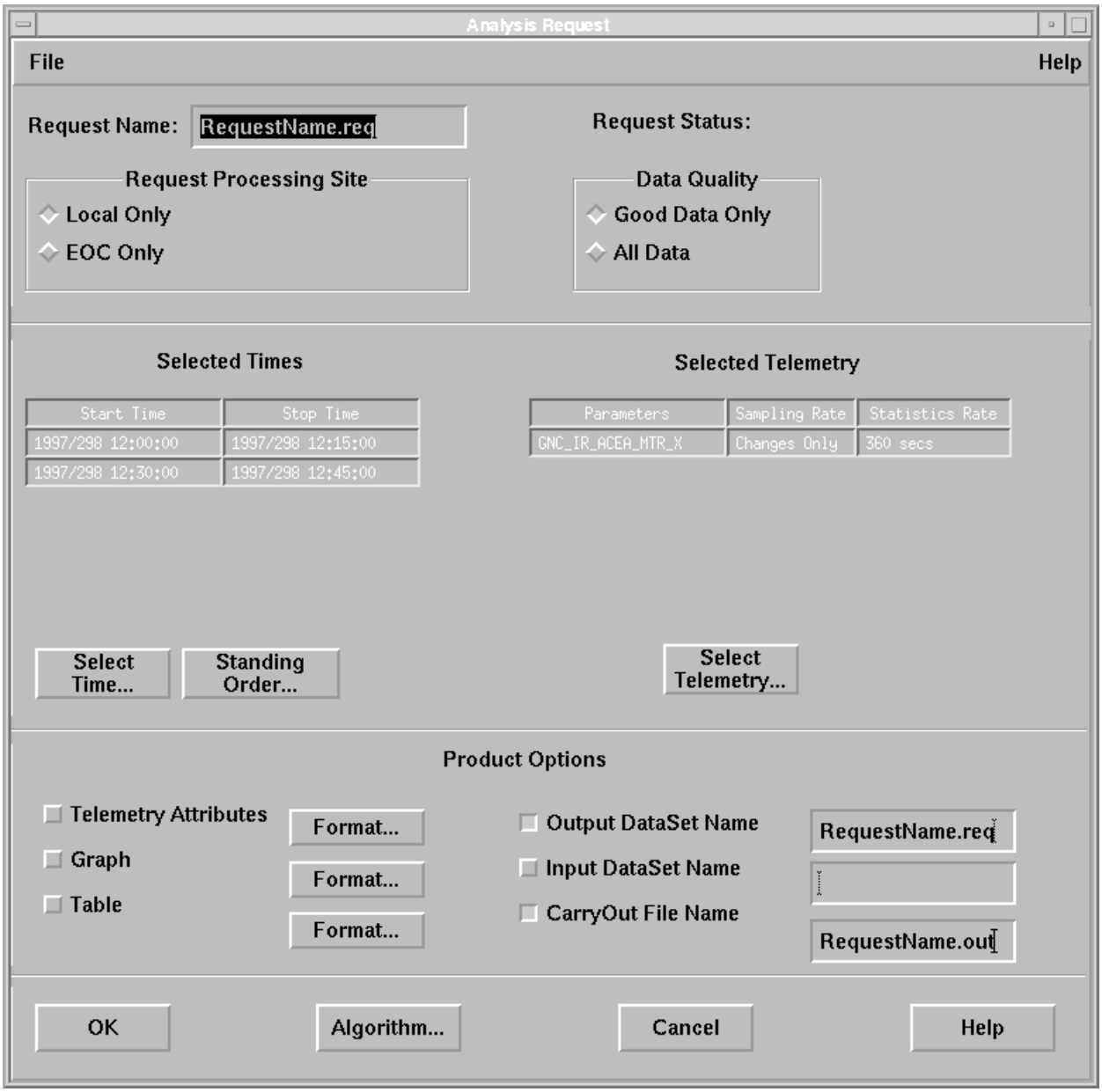

Figure 10.2.1-1. Analysis Request Window

3. Enter a request name in the Request Name text field. The naming convention is <request name>.req and can contain up to 56 characters, including the .req extension.
4. To select start and stop times for the data to be analyzed:
   a. Click Select Time. This opens the Selected Pair Times window (see Figure 10.2.1-2).

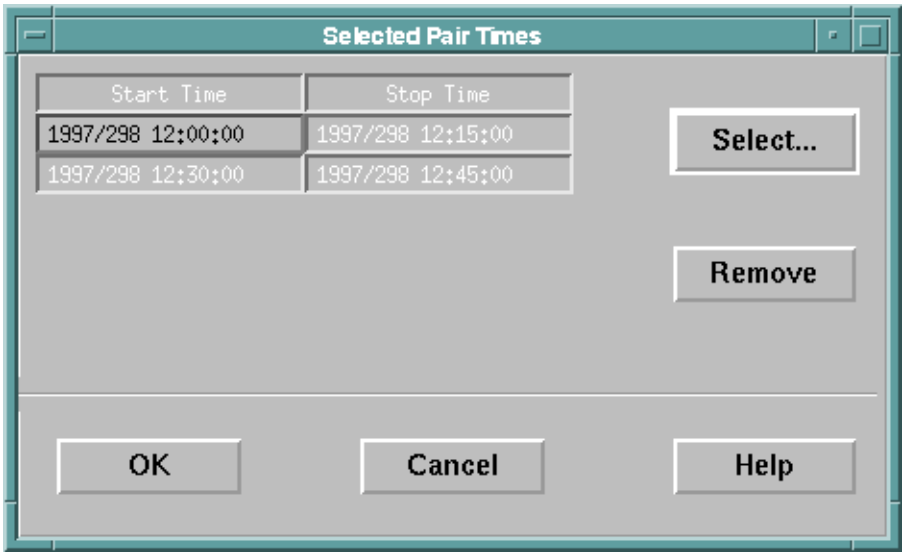

Figure 10.2.1-2. Selected Pair Times Window

b. Click Select to enter start/stop times. The Pair Time Selector display will open (see Figure 10.2.1-3).

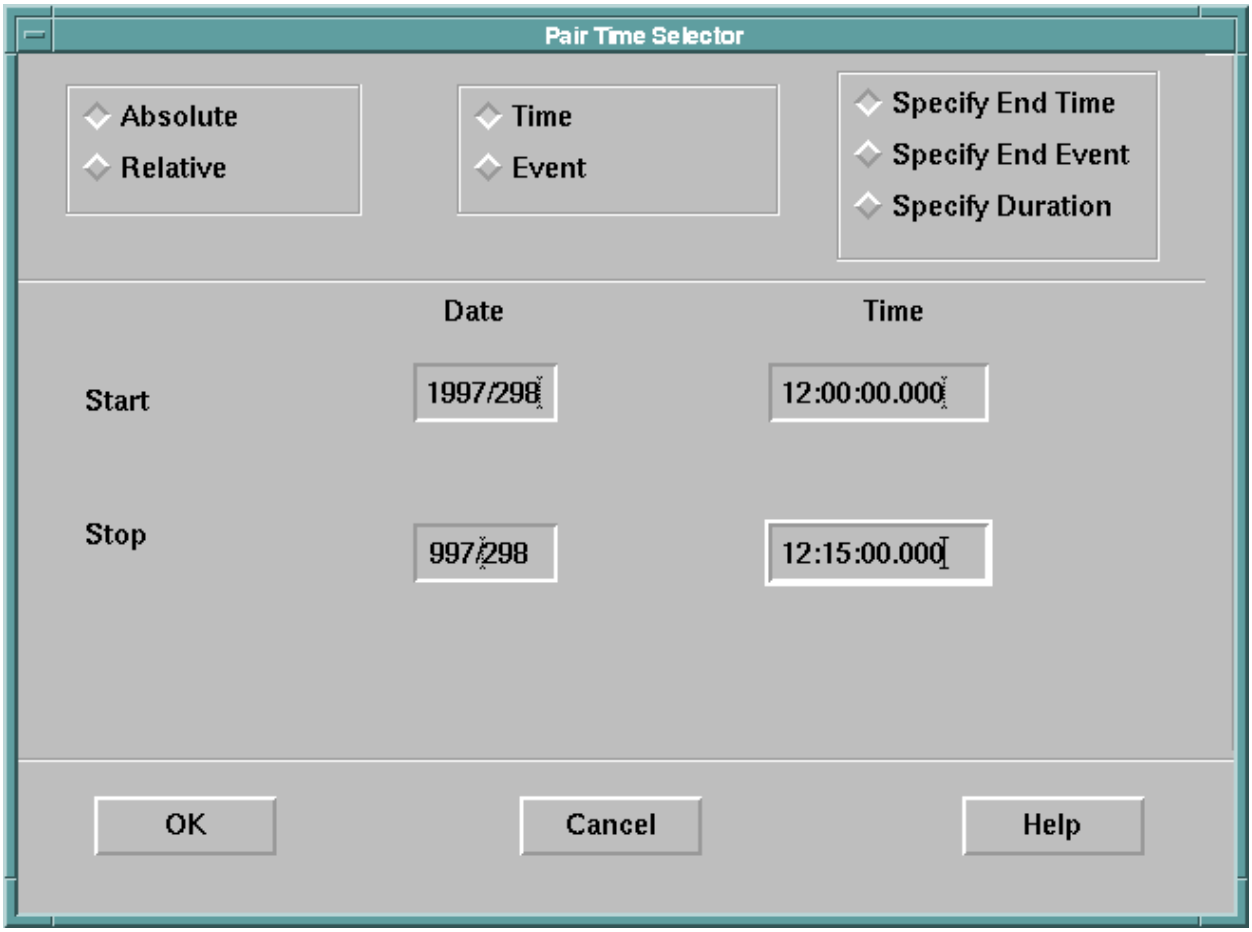

Figure 10.2.1-3. Pair Time Selector Window

c. Modify if necessary, the start date (YYYY.DDD); press <Return> to accept this new start date.

d. Modify if necessary, the start time (HH:MM:SS.mmm); press <Return> to accept this new start time.

e. Modify if necessary, the end date (YYYY.DDD); press <Return> to accept this new end date.

f. Modify if necessary, the end time (HH:MM:SS.mmm); press <Return> to accept this new end time.

g. Click OK to accept this pair time to be used in the analysis request. The Pair Time Selector window will close and the user will return to the Selected Pair Times window. Select another pair time if necessary by repeating steps b through g. The user can also delete any pair times generated by highlighting the time and clicking Remove.

h. On the Selected Pair Times window click OK. The pair time is now selected and displayed in the table labeled Selected Times on the Analysis Request Builder window.

NOTE

The **Absolute** and **Relative** selection boxes will invoke different dialog boxes. The **Time** and **Event** selection boxes may be user-defined as relative or absolute. Examples of absolute and relative time and events include: absolute time - February 14, 1997; relative time - the first day of every month; absolute event - sunrise on February 12, 1997; relative event - sunset every first day of the month.

5. To select a parameter with sampling and statistics rates:

a. Click Select Telemetry to open the Analysis Telemetry Selector window (see Figure 10.2.1-4).

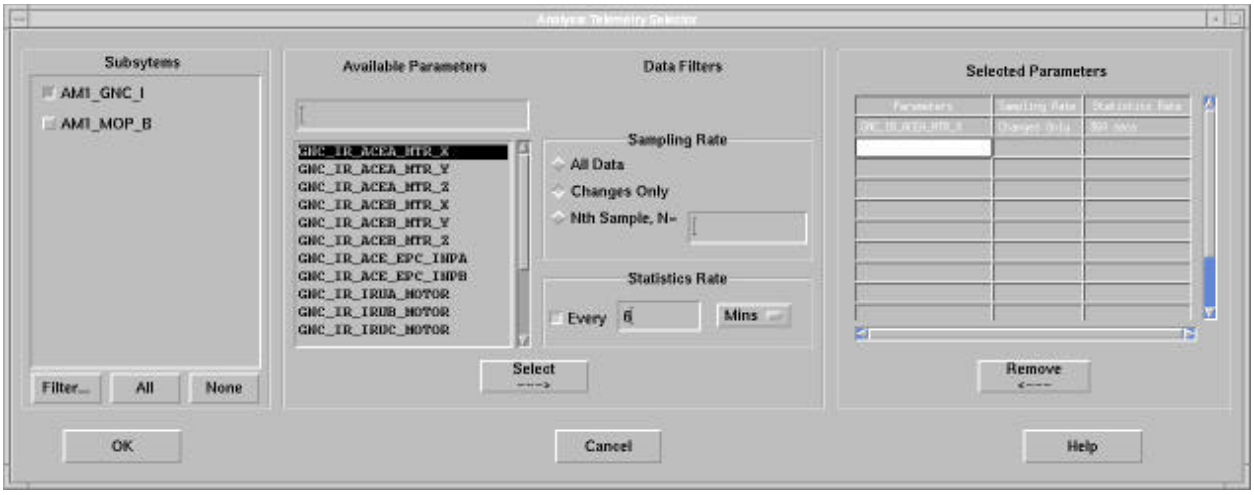

Figure 10.2.1-4. Analysis Telemetry Selector Window

b. Select a subsystem path.

1. Click Filter to open the Selection Filter window (see Figure 10.2.1-5).

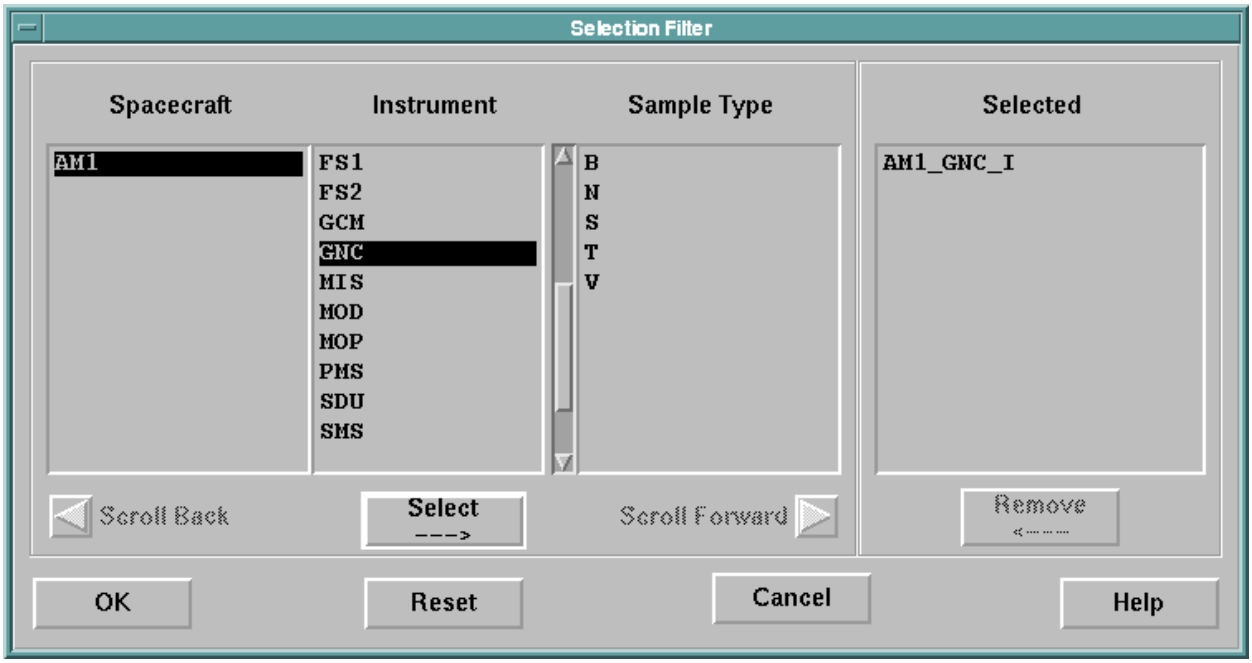

Figure 10.2.1-5. Selection Filter Window

2. Click a spacecraft subsystem (e.g., AM1) from the Spacecraft list.

3. Click on an Instrument/Subsystem (e.g., GNC) from the Instrument list. For Instrument / Subsystem definitions, see the DATA FORMAT CONTROL DOCUMENT for the Earth Observing System Flight Operations Segment Project Data Base, reference 505-10-35.

4. Click on a sample (e.g., **I**) from the Sample Type list. For Telemetry Sample Type definitions, see the DATA FORMAT CONTROL DOCUMENT for the Earth Observing System Flight Operations Segment Project Data Base, reference 505-10-35.

5. Click Select to select this subsystem mnemonic. The subsystem mnemonic ("AM1_GNC_I") is now selected and displayed in the right most list box labeled Selected.

6. Select another subsystem mnemonic, if necessary, by following the instructions indicated in steps 2 through 5.

7. Click OK to accept the selected subsystem mnemonics to be used in selecting parameters. The selected subsystem mnemonics are now displayed in the list box labeled Subsystems in the Analysis Telemetry Selector window.

    c. On the Analysis Telemetry Selector window, click on a mnemonic prefix (e.g., AM1_GNC_I) in the list box Subsystems. A list of available parameters for this subsystem mnemonic prefix is displayed in the listbox labeled Available Parameters.

    d. Click a parameter (e.g., a GNC_IR_ACEA_MTR_X) in the list box Available Parameters.

e. Select a sampling rate by clicking a radio button (e.g., changes only) inside the frame labeled Sampling Rate.

f. If desired, select a statistics rate by entering an integer in the Statistics Rate text field (e.g., 6) and selecting the Every toggle. Choose a rate in the pulldown Option menu (e.g., Mins).

g. Click Select to select parameter GNC_IR_ACEA_MTR_X with sampling rate Changes Only and statistics rate 6 minutes (or 360 seconds). These values are now selected and displayed in the table labeled Selected Parameters.

h. Repeat steps c through g to select additional parameters. At any time the user can delete any of the requests by highlighting an entry listed under the Selected Parameters and clicking Remove.

i. Click OK to accept the selected parameter(s) with sampling and statistics rates to be used in the Analysis Request. These are now displayed in the table labeled Selected Telemetry in the Analysis Request Builder window.

8. On the Analysis Request Builder window, click the Output DataSet Name toggle. Enter the file name in the text field to write the results of the analysis request. Use the same <request name>.**req** as the output dataset name. Click the CarryOut File Name toggle. The carryout file name uses the same name as the request except with a *.out extension. The Input DataSet Name uses an existing dataset file instead of telemetry for replay. All these fields can be up to 56 characters, including the extension.

9. Save this request by clicking the menu option Save As under the menu option File located on the top menu bar. Enter the name of the file to be saved and click OK.

10. To submit this request, click OK on the Analysis Request window. The Analysis Request Builder window closes. To modify a request, bring up the Analysis Request window from the Tools menu as in step 1 of section 10.2.1. Select the Open option from the File menu pull-down. A file selection dialog will prompt users to select a file. Enter a file name and click OK. The selected Analysis Request will appear in the Analysis Request window. Each Analysis Request will result in the creation of one or more logical strings. Users should not shutdown user stations with active logical strings associated with Analysis Requests. Please refer to Section 9.2 for more information on logical strings.

10.2.2 Adding User-Defined Algorithms to an Analysis Request

User-Defined Algorithms are C/C++ algorithms written by users and dynamically linked to the analysis request process. User algorithms can take up to twenty (20) input parameters and output up to twenty (20) parameters into the dataset generated by the analysis request. These output parameters can be placed in reports, tables, and graphs like other parameters.

Since FOS software can run on multiple platforms, and analysis requests can be farmed out to any of these platforms, the process of compiling user-defined algorithms will be a configuration controlled procedure, requiring platform specific compiler flags.

1. **Sample User-Defined Algorithm written in C++, with explanatory comments**:

```
#include"EcTypes.h" // include file used to define User Algorithm specific data types

/* the following variables map to input and output parameters (mnemonics) of the algorithm and MUST be

global */

/*INPUTS #a required comment needed by the User Algorithm Process. Distinguishes input and output

parameters */

EcTReal SolarArrayCurrent1; // an input. EcTReal is an algorithm data type for floating point numbers

EcTReal SolarArrayCurrent2; // another input

EcTInt SolarArrayCount; // and integer data type input

//OUTPUTS # another required comment, indicating the remaining variables map to outputs

EcTReal TotalCurrent; // a floating point output

EcTInt AnInteger; // and integer output

EcTBoolean AnInteger_OutputComplete = EcTFalse; /* an output flag telling the algorithm software whether to output the value of AnInteger. Useful for performing running sums where the algorithm may execute every second but results are only needed every ten minutes. */

extern "C" UserFunction() /*  algorithms must be extern "C", return void, and take no arguments. The function name must be the same as the filename minus extension ie. UserFunction.C would be the filename for this algorithm */

{

static int count = 0; // local variables need not be EcTInt/EcTReal

// compute total solar array current and store in output

TotalCurrent = SolarArrayCurrent1 + SolarArrayCurrent2;

// now perform a meaningless example of a running sum

count += SolarArrayCount;

if (count > SOME_BIG_NUMBER)

{

    AnInteger = count;
```

```
        AnInteger_OutputComplete = EcDTrue; /* tell the software to output
        AnInteger */
        count = 0;
    }
    else
    {
        AnInteger_OutputComplete = EcDFalse; /* tell the software NOT to output
        AnInteger */
      }
    }
```

2. **User-Defined Algorithm Registration**. Once a user-defined algorithm is compiled and copied to the appropriate CM area (operational, test, or training), it must be registered. This involves opening the Algorithm Registration window (see Figure 10.2.2-1) from the Tools menu, selecting the appropriate algorithm, and clicking Register. Selecting an algorithm can be accomplished by typing the name of the algorithm into the Source File Name text field or by clicking Select File and selecting an algorithm name from the file selection menu. FOS software will verify the format of the algorithm source file and report either success or a list of errors in the Status text box. The Algorithm Description text box allows the user to give a description of the registered algorithm.

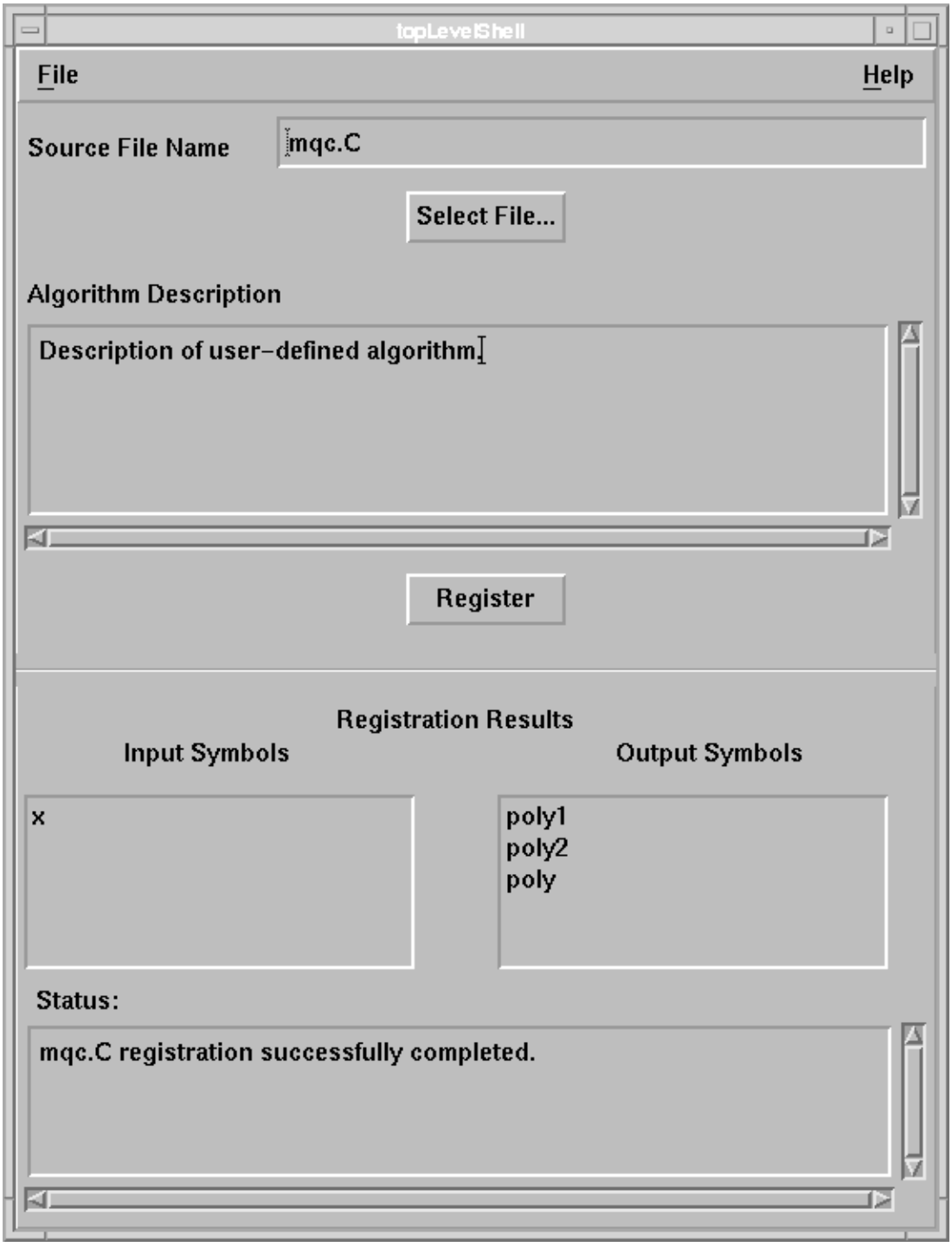

Figure 10.2.2-1. Algorithm Registration Window

3. **Adding User-Defined Algorithms to an Analysis Request**. From the **Tools** menu select the **Analysis Request Builder** (see Figure 10.1.1-1). Choose Algorithms from the Analysis Request Builder display to open the Algorithm Request window (see Figure 10.2.2-2).

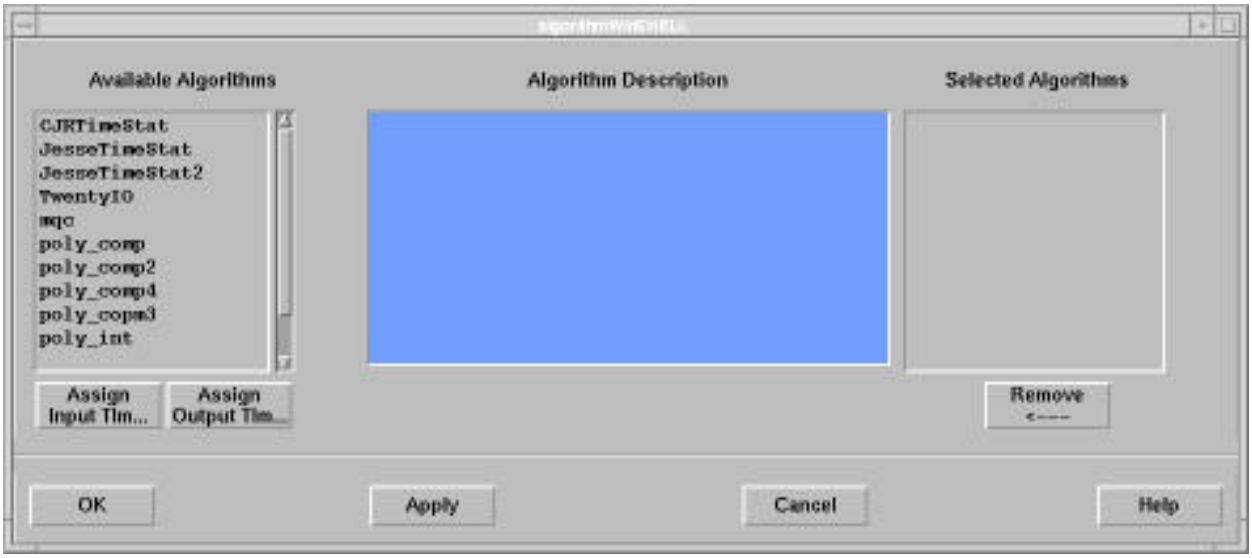

Figure 10.2.2-2. Algorithm Request Window

From the Algorithm Request display, select the algorithm to be added. At this point the user must make several selections before clicking **OK**. The selections include adding input and output symbols in the algorithm source file that are mapped to input and output telemetry parameters. First click **Assign Input Tlm** to open the Input Parameter Mapping window (see Figure 10.2.2-3).

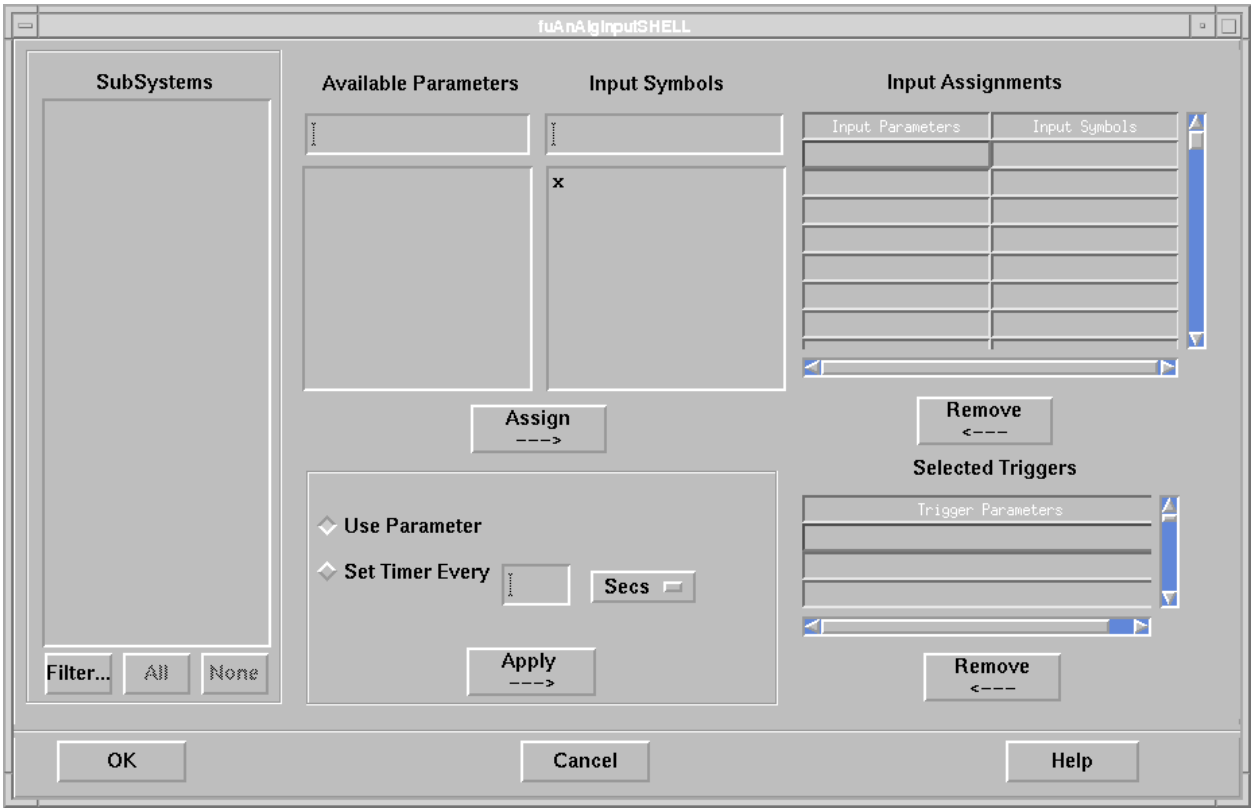

Figure 10.2.2-3. Algorithm Input Parameter Mapping Window

Select the desired telemetry parameters to be mapped to the input symbols and click Assign. Next, choose an algorithm invocation timing mechanism that allows the user to specify timing rates, parameters, and triggers. This will determine exactly when and how often the algorithm is invoked. The user may select a timer value, that will cause the algorithm to be invoked every N seconds, or select telemetry parameters that trigger the algorithm execution. All parameters selected as triggers will update before the algorithm is invoked. Parameters selected as triggers need not be part of the algorithm. For example, the packet ID parameter could be used as a trigger parameter to cause the algorithm to update once every packet, even though the packet ID parameter is not used in the algorithm. Click Apply to accept the desired timing mechanism, then OK. The user returns to the Algorithm Request display.

From the Analysis Request window, select **Assign Output Tlm**. This opens the Output Parameter Mapping window (see Figure 10.2.2-4) that allows the user to map telemetry parameters to output symbols. Select the desired output parameters and symbols and click Assign. After the desired parameters and symbols are selected, click OK. Again the user returns to the Algorithm Request window.

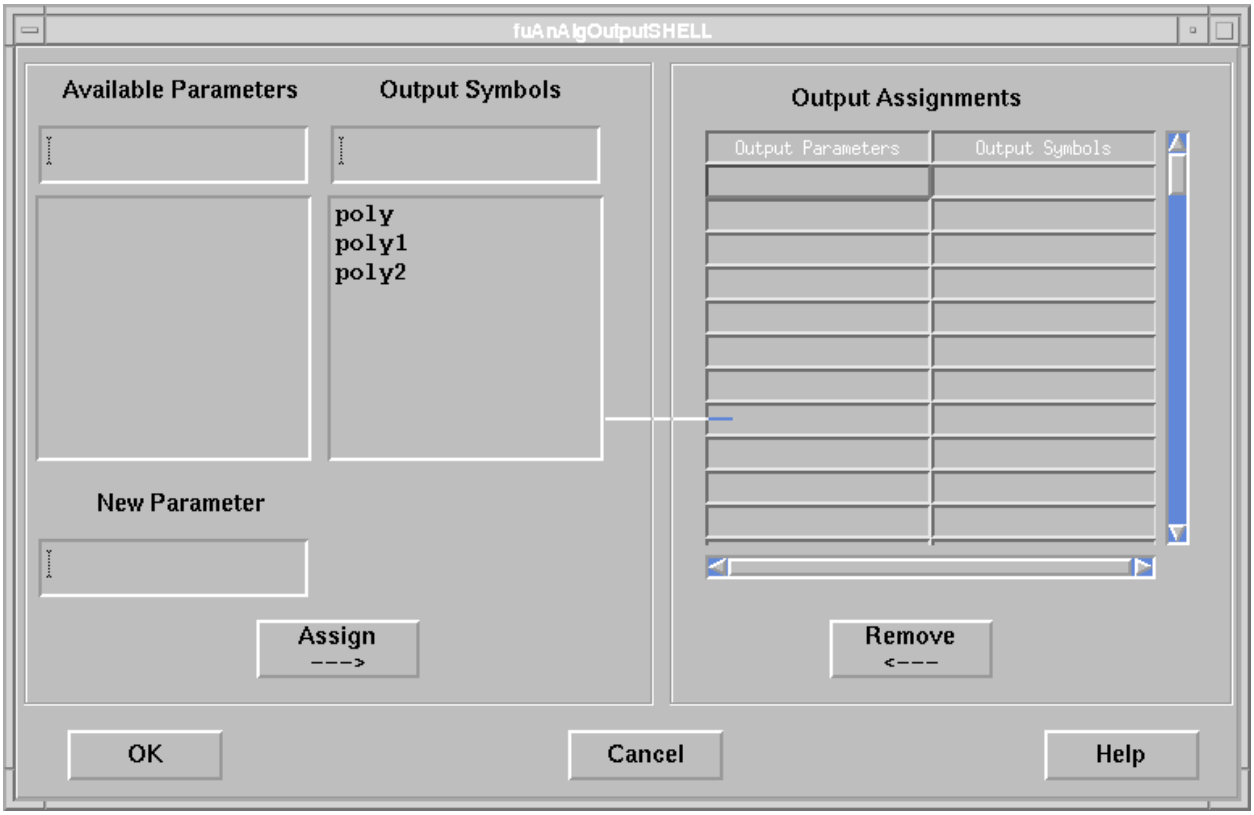

Figure 10.2.2-4. Algorithm Output Parameter Mapping Window

After selections have been applied, the algorithm name will appear in the Selected Algorithm list box on the Algorithm Request window. This allows the user to re-select the algorithm for modifications to the input/output mapping or the invocation rate. Algorithms in the Selected Algorithms list can also be selected for deletion. This is done by clicking Remove from the Algorithm Request window. When the desired selections are ready for processing, the user MUST click Apply from the Algorithm Request window to associate the selected inputs and outputs with the selected algorithm. Click OK, this takes the user back to the Analysis Request window. Note that once an algorithm is selected, the necessary telemetry inputs and dataset outputs are automatically part of the Analysis Request, and do not need to be added via the Select Telemetry option of the Analysis Request Builder display.

10.3 History Processing

Analysis report functions can be used to access historical data based on archived spacecraft telemetry, ground telemetry and command parameters based on archive data files.

10.3.1 Spacecraft Telemetry

AM-1 spacecraft telemetry data types include housekeeping, health and safety, and standby. Telemetry can be replayed in shared or dedicated fashion and is used to support Analysis Request processing. Telemetry data are stored in hourly files using the following naming convention:

spacecraft + spacecraft time + "." + data type + channel

where:

spacecraft = AM1.

spacecraft time = Spacecraft time, Julian format to the hour (yyyydddhh).

data type = HK, HS, SB.

channel = I or Q (Applies to a real-time contact only; Does not apply to rate-buffered data).

So, the file AM1199810010.HKI would represent the I-channel, real-time AM-1 housekeeping data for spacecraft time of: year 1998, day 100, and hour 10. These telemetry files are located in the directory build as follows:

TLM_PATH

This directory is an environment variable that can be reconfigured by the Database Administrator. For example: /net/beeper/fosb/int/AM1

10.3.2 Ground Telemetry

Ground telemetry data types include NCC user performance data, NCC GCM buffers, and EDOS CODAs. The data are stored in files named as follows:

spacecraft + ground time + "." + data type

where:

spacecraft = AM1

ground time = ground receipt time for NCC buffers, embedded time for EDOS CODAs

data type = GCM, UPD, EDOS

So, the file AM1199830015.EDOS would contain the EDOS CODAs received during 1998, day 300, and hour 15 for AM-1. The files are located in the same directory as the files for spacecraft telemetry (refer to Section 10.3.1).

a. **NCC Performance**. NCC User Performance Data can be replayed or processed in the same way as spacecraft telemetry data.

b. **NCC Configuration changes**. EOC Ground Control Message Requests sent to the NCC to change SN real-time configuration can be replayed or processed in the same way as spacecraft telemetry data

c. **EDOS CODA**. EDOS CODAs can be replayed or processed in the same way as spacecraft telemetry data.

10.3.3 Command

Command data types include Command Link Transmission Units (CLTU)and Command Link Control Words (CLCW). The data are stored in files named as follows:

spacecraft + ground time + "." + data type

where:

spacecraft = AM1

ground time =

data type = CLTU, or CLCW

So, the file AM1199830015.CLCW would contain the CLCWs received during 1998, day 300, and hour 15 for AM-1. The files are located in the same directory as the files for spacecraft telemetry (refer to Section 10.3.1).

10.4 Database Management

The FOS Data Management Subsystem (DMS) uses a database to store and manipulate data to support ECS mission operations. A Sybase server provides the foundation for the FOS Database. The installation of the FOS Database is described in the FOS EOC Installation Guide. FOS created three databases unvalidated (am1_fos_unv), validated (am1_fos_val) and the operational database (am1-fos_ops) to aid in PDB processing and real-time operations.

A member of the FOT will be designated as the Database Administrator (DBA). The DBA will have prime responsibility for overseeing the FOS Database and its related operations. A user interface for database operations is available through World Wide Web WWW pages. The DBA has access to the options shown on the Database Utilities page shown in Figure 10.4-1. All other FOT members have access to options shown on the Database Utilities (User) page shown in Figure 10.4-2.

10.4.1 Database Administration

The FOS DBA accesses the FOS Database through the *fos_dba* Sybase account. This account owns all objects within the FOS Database and therefore has all privileges to the data. The DBA's .cshrc file sources two Unix scripts to set environment variables to access the FOS Database.

The *fos_sybsetup.script* sets the Sybase parameters and other environment variables used to interface to Sybase. This file is modifiable only by the System and/or Database Administrators. Any users accessing the FOS Database need to source this script in their .cshrc file.

The *fos_dba_env.script* sets the environment variables for the DBA account. This file is readable/modifiable only by the System and/or Database Administrators. Whenever the administrator modifies the *fos_dba* Sybase account password, a modification to this file is required.

10.4.2 PDB Processing

The PDB generation process is performed by the DBA in a non-real-time, interactive environment. This process provides definitions needed to support mission operations. To ensure successful processing, you should verify proper configuration of the environment. The environment variable, $PDB_DIR, must be configured to the PDB operational directory. In general, all database scripts must be run from their designated directory for the AM1 mission (e.g., /fos/ops/am1/db).

The PDB generation process begins with the transfer of the PDB files to the dedicated area, $PDB_DIR/input, at the EOC. These files are derived from the most recent version of the Integration & Test (I&T) database supplied by the spacecraft contractor. These definition files, which contain telemetry and command information, will be loaded into the FOS unvalidated Database $UNV_DB, by clicking **PDB load** on the FOS Database Utilities page. Each new release of the I&T database supersedes the previous version However, FOT entered data will marked with a pdb source of FOT and will not be deleted or overwritten by the new version. During this process, input files are moved to a newly created subdirectory and named according to the I&T version of the input files.

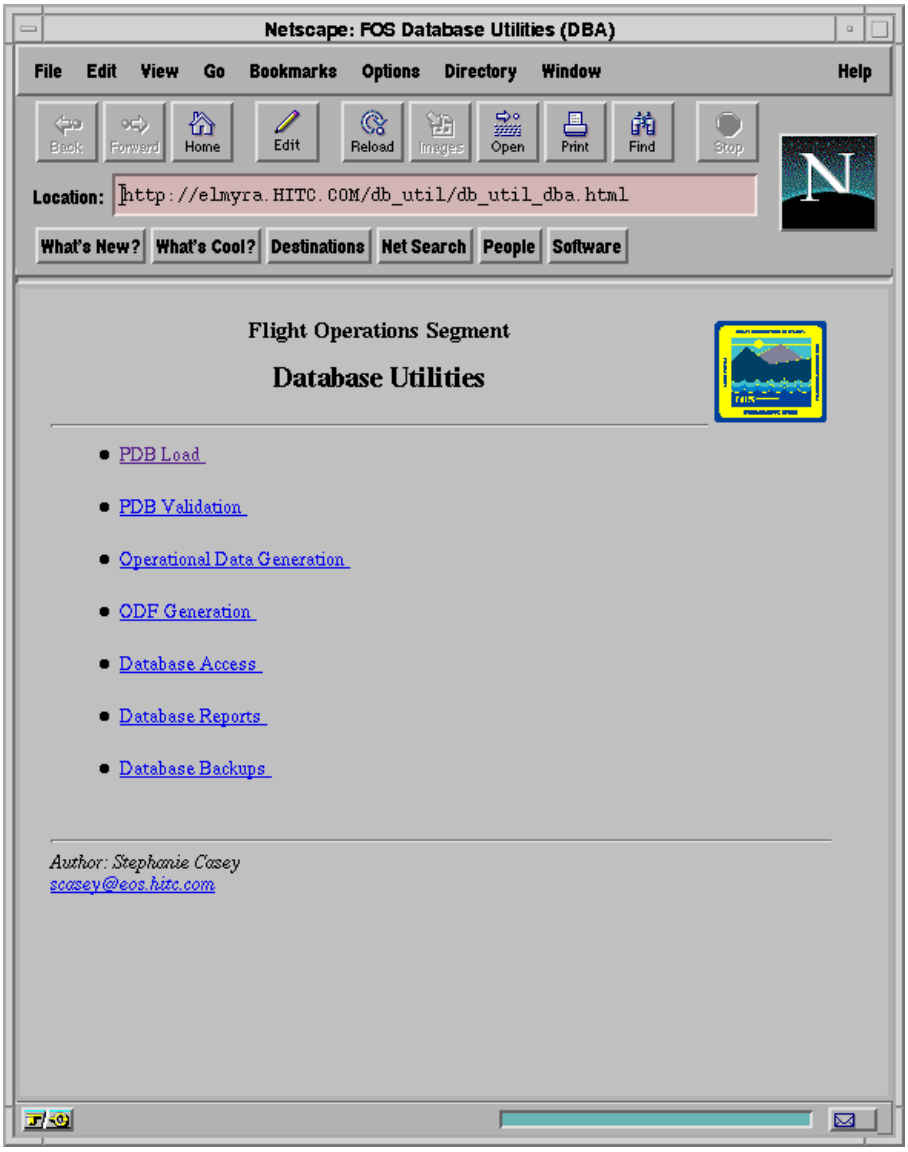

Figure 10.4-1. Database Utilities (DBA) Page

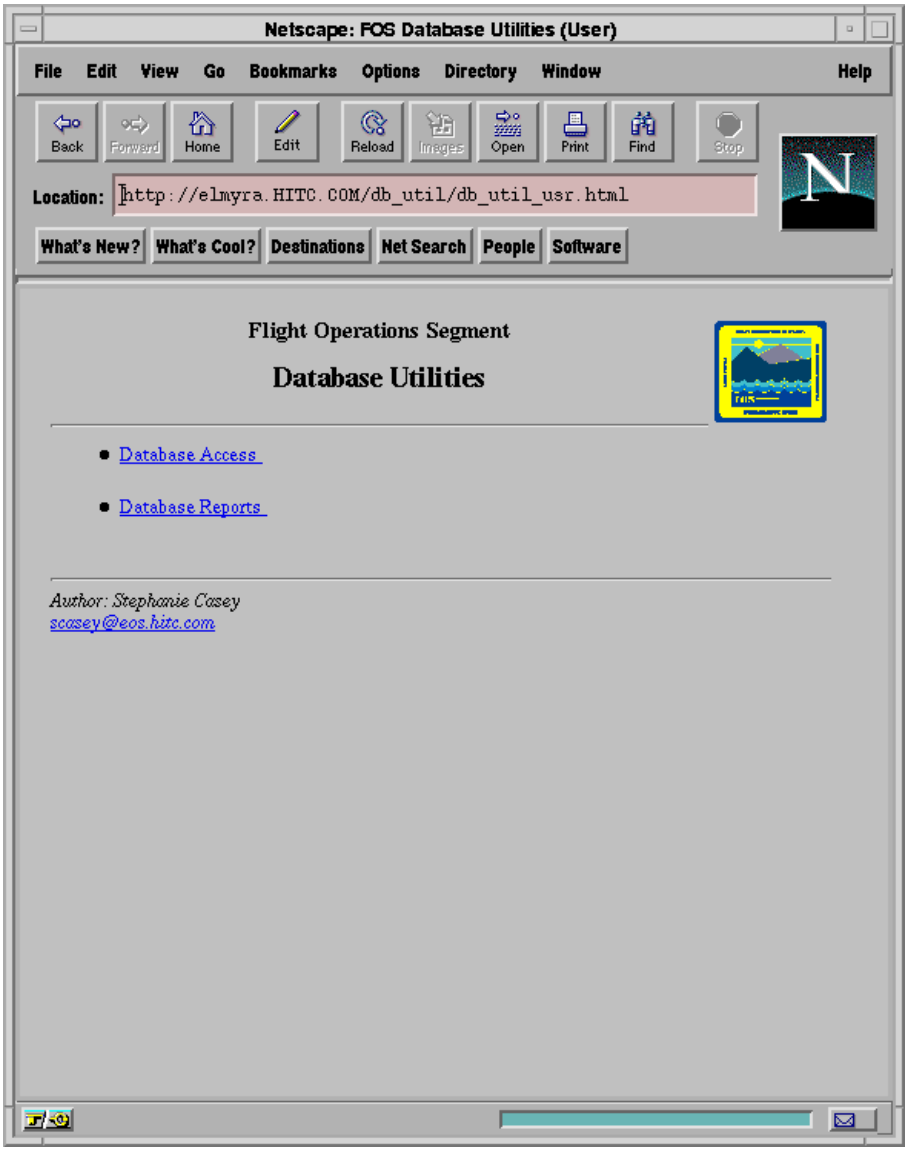

Figure 10.4-2. Database Utilities (User) Page

Validation of PDB definitions ensures the accuracy of information used to support mission operations. It is a necessary step which includes syntax checking, verification of values, and cross-checking of related definitions contained in the unvalidated database. Valid records are then passed in the validated database. Additionally, due to dependencies between PDB data types (i.e., command definitions contain telemetry parameters), a specific order must be enforced when validating these definitions. PDB validation is invoked by clicking **PDB Validation** on the FOS Database Utilities page. The validation options are displayed on the PDB Validation page (see Figure 10.4.2-1). Validate the PDB will perform telemetry and command definition validation. Once this portion of the PDB is validated, the constraints and activities are validated. Validate PDB Constraints Only may be selected when new command constraints are introduced to a previously validated database. This validation is dependent on a valid command definition.

Validate PDB Activities Only may be selected when new activities are introduced to a previously validated database.

During the validation process, data is moved into the valid database within the FOS database. Upon completion, a validation summary is available for the DBA's review by clicking **View the PDB Validation Output File** on the PDB Validation page. This summary report is vital to determining problems with PDB definitions.

Once validation is complete and the report is reviewed. The DBA determines if the operational database tables should be generated. If so, the DBA should select Return to the Database Utilities screen.

Operational database tables are generated by clicking **Operational Data Generation** on the FOS Database Utilities page. Operational data generation options include generating a new operational database, generating a new version of constraints only, or new activities only (see Figure 10.4.2-2). Valid information is moved from the valid database $VAL_DB into the operational database $OPS_DB within the FOS database and a set of dump files are created, these files generate PDB output distribution files.

The operational database file generation process creates the input files used by various FOS subsystems. Each ODF build operation creates one or more output files in the output directory $ODF_DIR (refer to Table 10.4.2-1).

Table 10.4.2-1. ODF Source Generation

EXECUTABLE	ODF
FdDbBuildAnlOdf	AnalysisOdf_$ODFVERSION
FdDbBuildCmsOdf	CommandOdf_$ODFVERSION
FdDbBuildEventOdf	EventODF_$ODFVERSION
FdDbBuildFuiCmdOdf	FuiCommandODF_$ODFVERSION
FdDbBuildFuiOdf	ParmDataOdf_$ODFVERSION
FdDbBuildNccOdf	GcmrOdf_$ODFVERSION
FdDbBuildPidsOdf	TlmSelectFilterOdf_$ODFVERSION
	CmdSelectFilterOdf_$ODFVERSION
	PidsOdf_$ODFVERSION
FdDbBuildSysOdf	SystemOdf_$ODFVERSION
FdDbBuildTlmOdf	TlmDecom Odf_$OdfVersion
	TlmDumpOdf_$OdfVersion
FdDbBuildActivityOdf	ActivityOdf_$ODFVERSION
FdDbBuildCmdConOdf	CmdConOdf_$OdfVersion
FdDbBuildCmdFormatOdf	CmdFormatOdf_$OdfVersion
FdDbBuildCodaOdf	CodaOdf_$OdfVersion
FdDbBuildDetStringOdf	DetStringOdf_$OdfVersion
FdDbBuildFuiTlmMnemOdf	FuiTlmMnemOdf_$OdfVersion
FdDbBuildMemoryMaskOdf	MemoryMask_$OdfVersion
FdDbBuildMnemToPidOdf	MnemToPidOdf_$OdfVersion
FdDbBuildUpdOdf	UpdOdf_$OdfVersion

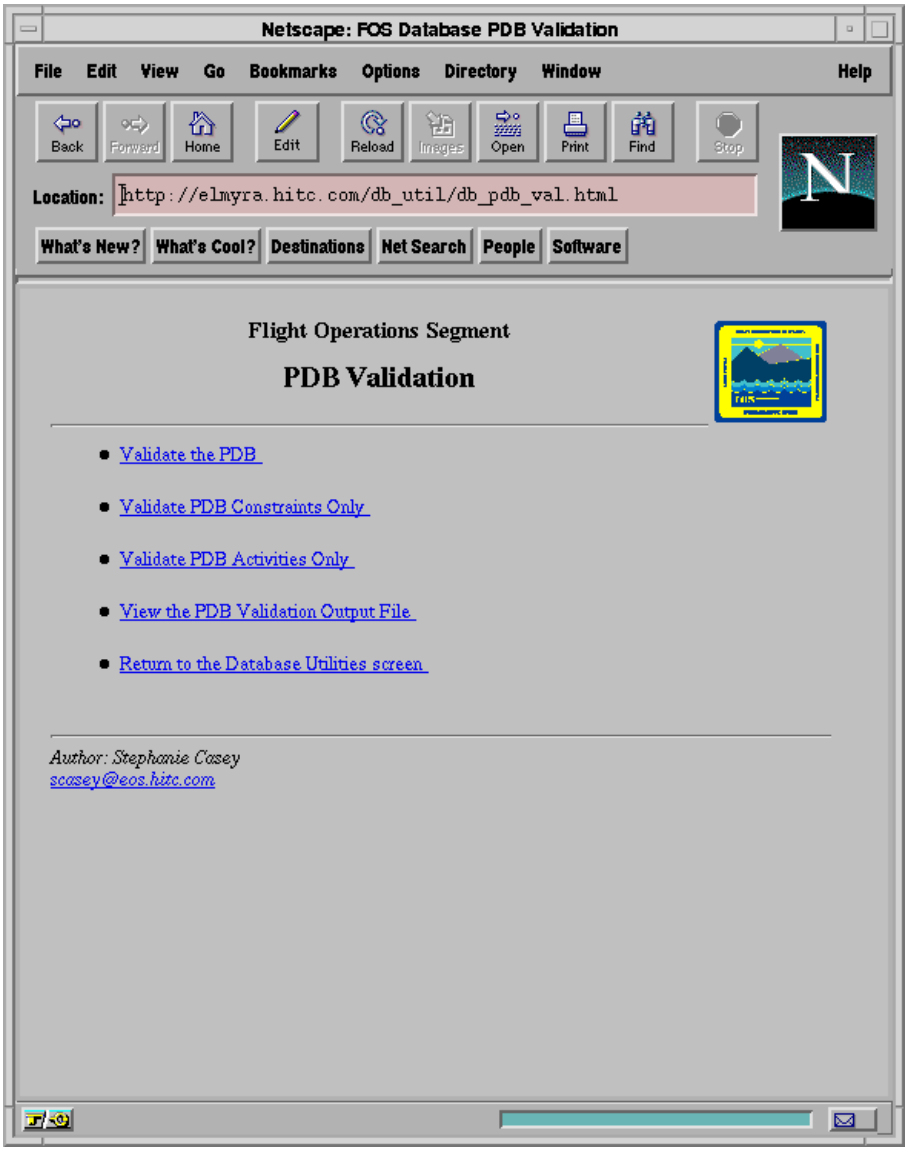

Figure 10.4.2-1. PDB Validation Page

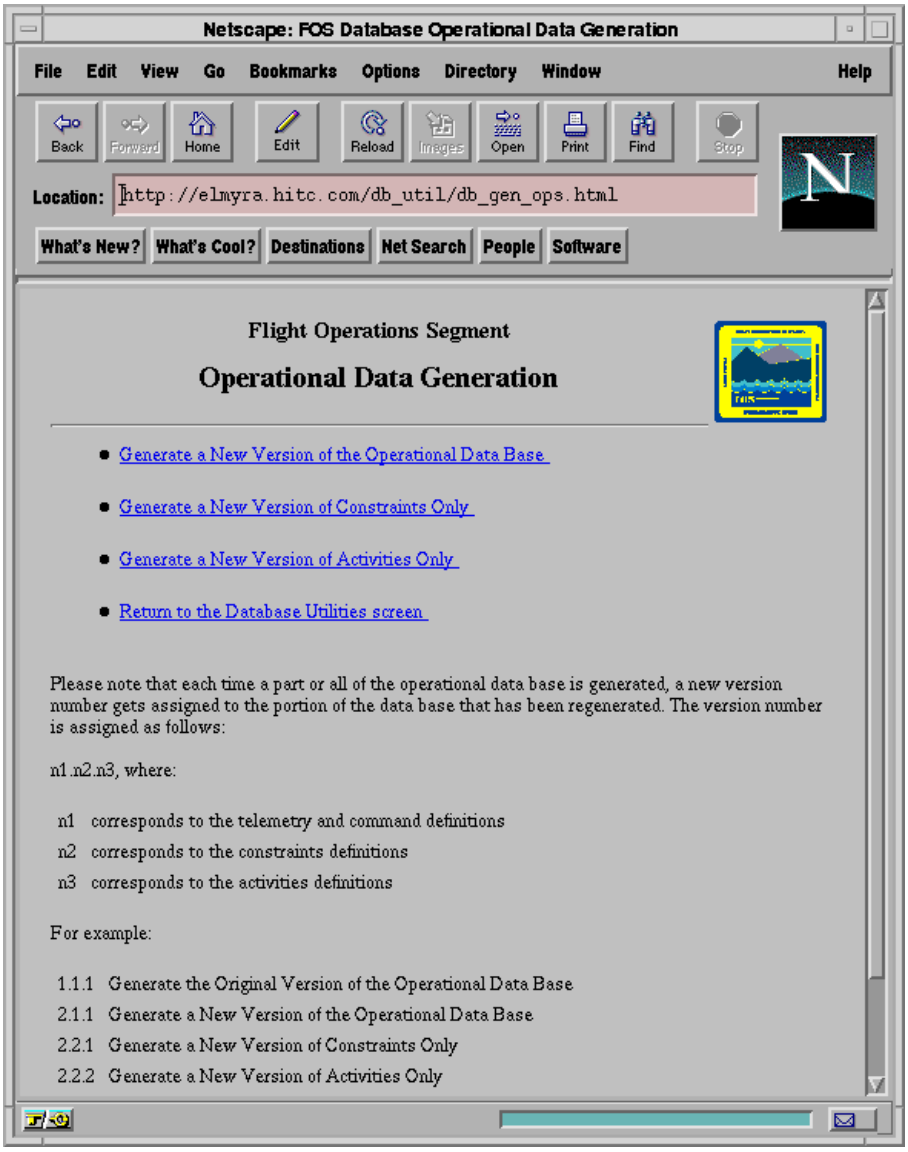

Figure 10.4.2-2. Operational Data Generation Page

10.4.3 FDF Products Processing

The FDF generation process is performed in a non-real-time, interactive environment. This process provides definitions needed to support mission operations. To ensure successful processing, you should verify proper configuration of the environment. The environment variable $FDF_DIR must be configured to the FDF operational directory. All database scripts must be run from their designated directory, such as /fos/ops/am1/db for AM-1.

The FDF generation process begins upon notification of the transfer of the FDF files to the dedicated directory, $FDF_DIR/input, at the EOC. All files transferred between FDD and EOC are conducted through File Transfer Protocol (FTP) (see Section 3.1.3 FDD Interface). The FDF

product files will be loaded into the FOS Database $UNV_DB. The FdFwFileWatcher process monitors the $FDF_DIR directory, when the File Watcher process encounters a signal file from FDD, the database script, fdf_load.script, will be invoked automatically. Each new release of the FDF database is appended to the existing database. During this process, input files are moved to a newly created subdirectory that reflects the time the files were received.

Validation of FDF definitions ensures the accuracy of information used to support mission operations. It is a necessary step which includes syntax checking and verifying of values for both the header and data files.

Files sent from FDF containing validation errors will be batched and returned to FDF for manual editing. No online editing capabilities are provided.

Validation reports for each FDF product will be available by clicking **Database Reports** on the Database Utilities page (refer to Section 10.4.5).

10.4.4 Database Access

The database access functions, allow the DBA to access the FOS database, is opened by clicking **Database Access** on the FOS Database Utilities page. The options are displayed on the Database Access page (see Figure 10.4.4-1). The user can access the FOS Database Utilities (User) page (see Figure 10.4-2).

The Database Access pages provide you with the ability to access data, however the DBA can manipulate data in the database as well, through Netscape.

At the bottom of the various database access forms is a **Clear Form** button that clears all previous entries and a **Submit** button that sends the request to the database. You may submit a request with any number of the fields entered. This will filter the search and return records that meet each of the field specifications. A form submitted without any entries will return all records in the database.

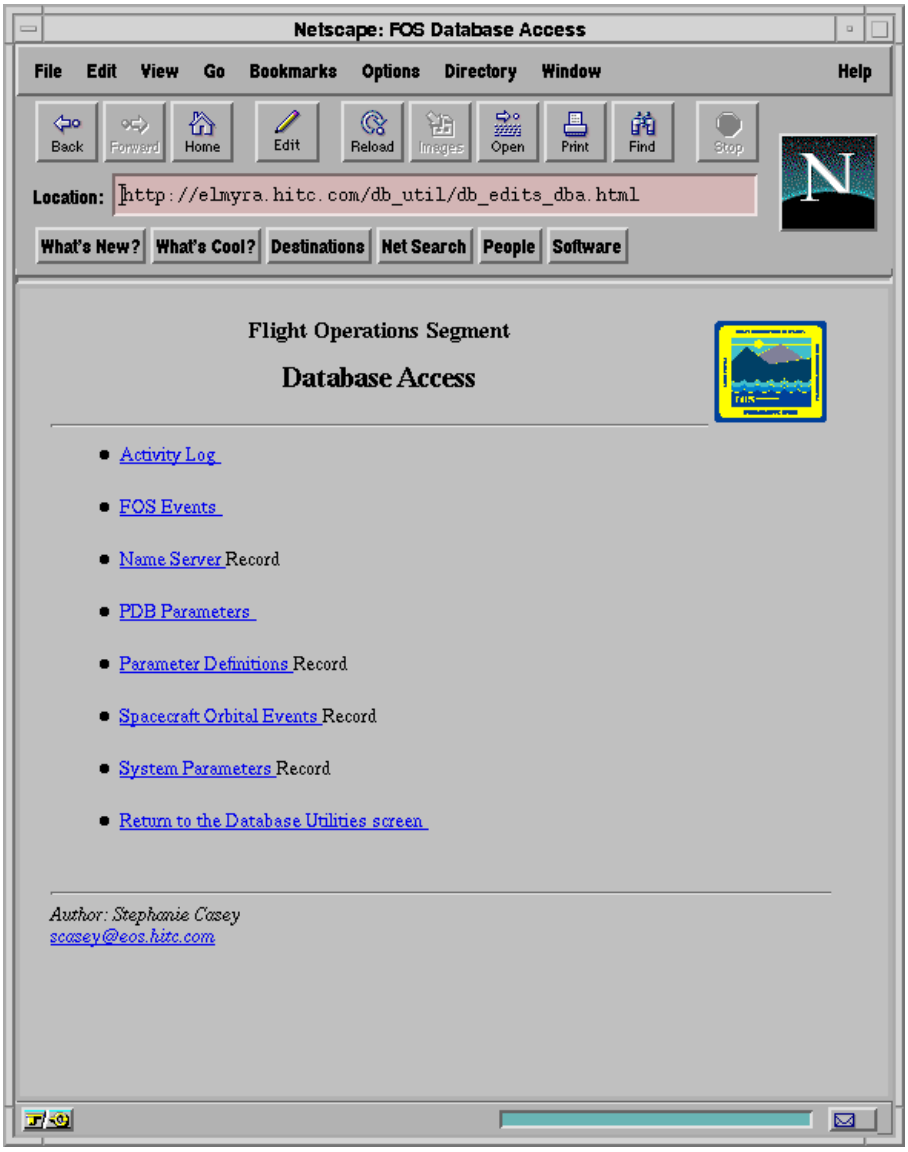

Figure 10.4.4-1. Database Access Page

10.4.4.1 Activity Log

The Spacecraft Activity Log Monitor process is begun when a string is started. The process will monitor telemetry for activity log messages. For each new activity log message received, an event is created. The event indicates whether the activity log was severe or informational, the activity log entry ID, the message associated with that ID, the time the entry was created, the cycle, and the five data words for the message. The Activity Log Monitor process also receives the dump data when the Activity Log table is dumped. When the monitor finishes processing the dump, one event message will be generated containing the number of new activity log messages and how many of those were severe.

All activity log messages are archived to the database and can be viewed through the Activity Log Web Browser.

Open the Activity Log Database Access page (see Figure 10.4.4.1-1) by clicking **Activity Log** on the Database Access page. The Activity Log page is used to retrieve spacecraft activities from the activity log archive.

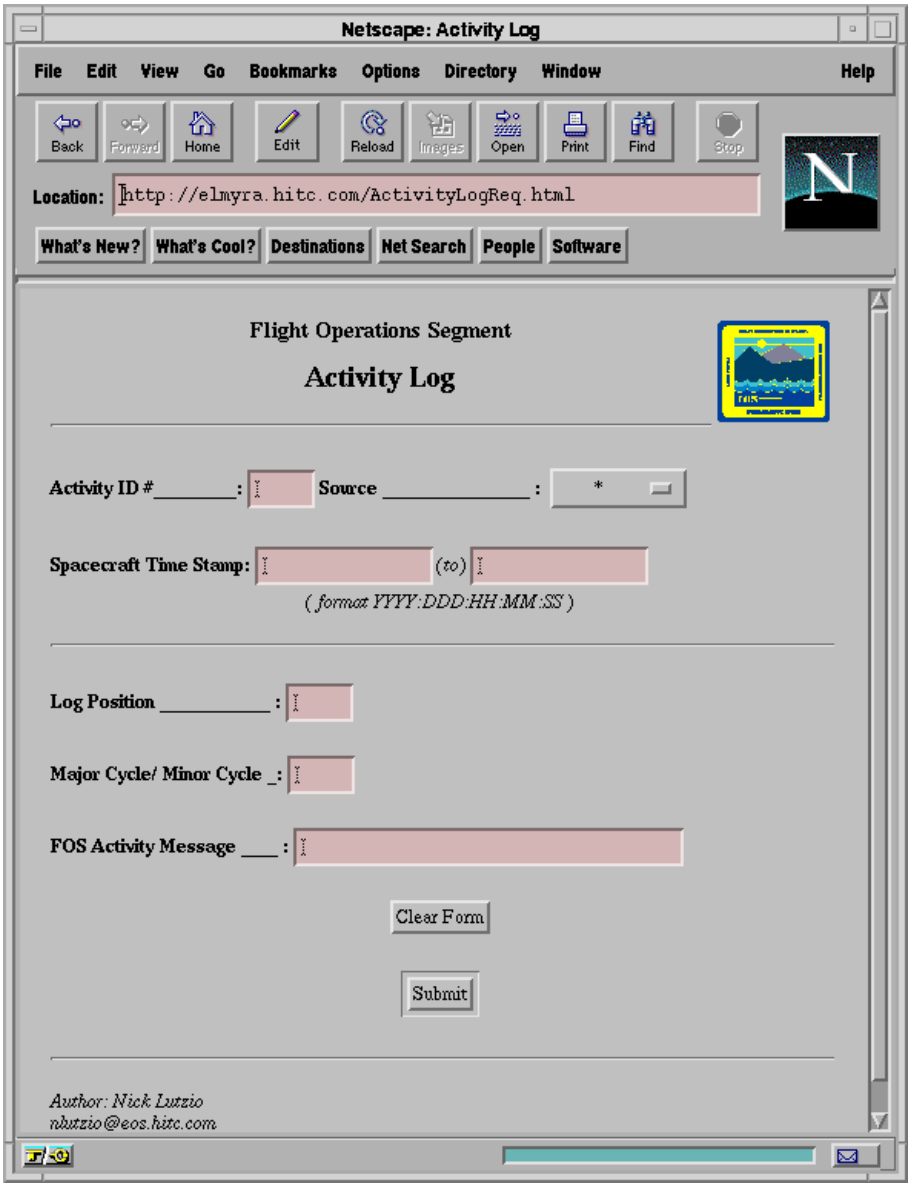

Figure 10.4.4.1-1. Activity Log Page

The following is a list of fields associated with the activity log along with a brief field description.

a. **Activity ID#**. Each activity has an associated identification number. Possible values for this field are 0 to 255.

b. **Source**. The source of data can either be real-time or simulation. This field also has ***** as an option. The asterisk will select activities from both sources.

c. **Spacecraft Time Stamp**. You have the ability to retrieve activities within a certain Time Stamp range. The format for this field is: yyyy:ddd:hh:mm:ss.

d. **Log Position**. The spacecraft contains 300 log positions to store activities before they are dumped. This field allows you to select which log position to filter on.

e. **Major Cycle/Minor Cycle**. Activities are trickled down with telemetry during each support.

f. **FOS Activity Message**. This field is used to retrieve any activities containing the string of characters entered into this field.

10.4.4.2 FOS Database Events

Open the FOS Database Events page (see Figure 10.4.4.2-1) by clicking **FOS Events** on the Database Access page.

A. **Event Definitions**. Open the Event Definitions page (see Figure 10.4.4.2-2) by clicking **Event Definitions** on the FOS Database Events page.

The Event Definitions page is used to define, modify, delete and view information stored in the Event ODF. If any changes are made regarding event definition, the user must generate the Event ODF to implement the changes. The following items are a list of fields associated with the event definitions along with a brief field description:

1. **Event Type**. The FOS subsystem that generated the event: ANA, CMS, CMD, DMS, FUI, PAS, RCM, RMS, TLM, or SYS.

2. **Severity**. The severity of the event. Valid severities are Fatal, Alarm, Warning, and Informational.

3. **Definition ID**. The event as a number. All integers are valid for this field.

4. **Event Name**. The field as a string. This field should follow the FOS naming convention for events (2-character subsystem ID, "CEv", Descriptive Name). An example of a DMS event is, FdCEvInvalidUserRequest.

5. **Background Text**. The background text stored in the Events Definitions file. This field should contain %s where a string needs substitution, a %d where an integer needs substitution, and a %t where time needs substitution.

   An example of a background text is "Printer %s Failed". The %s is filled by the application generating the event.

6. Event **Trigger**. The name of a procedure or executable associated with an event. For most events this field will be blank.

7. **Processing Route**. The name of the process event that needs to be routed. For most events this field will be blank.

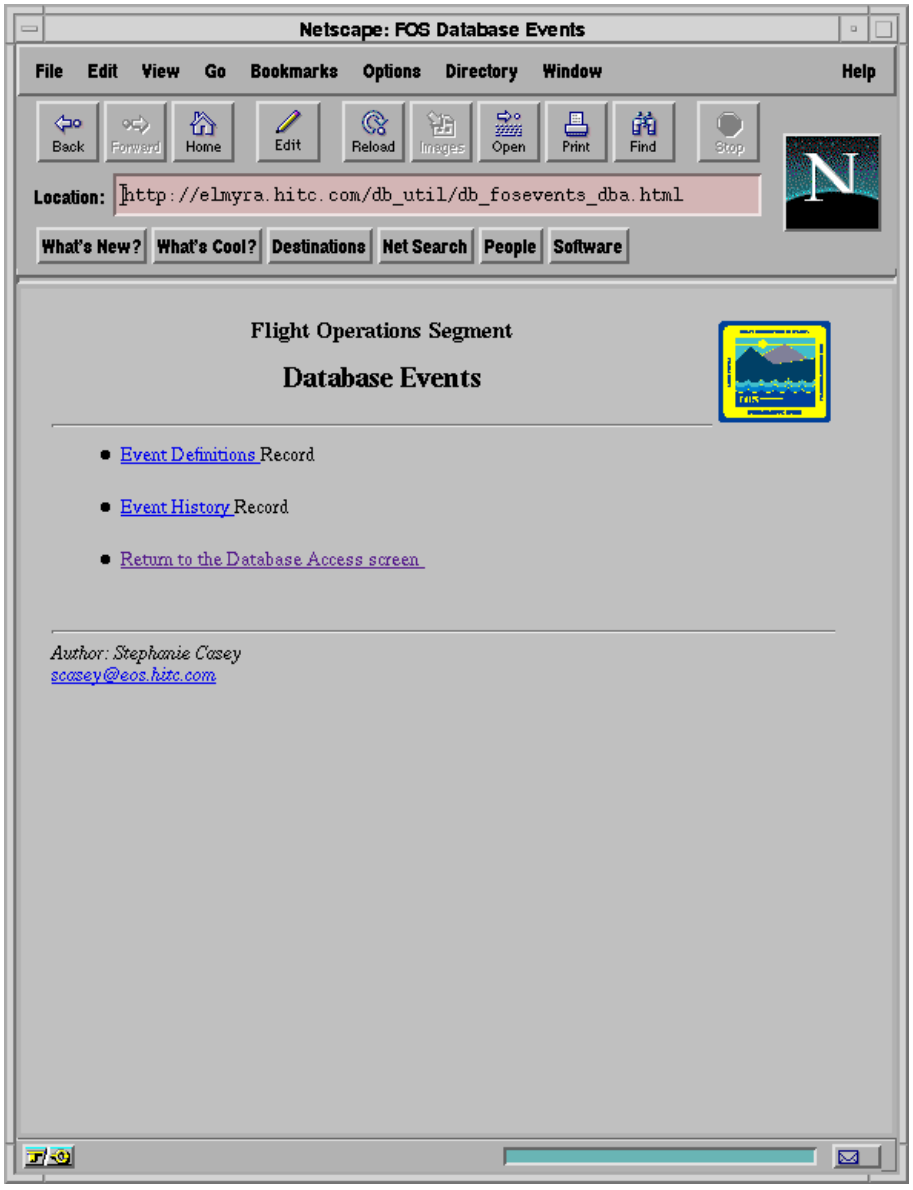

Figure 10.4.4.2-1. FOS Database Events Page

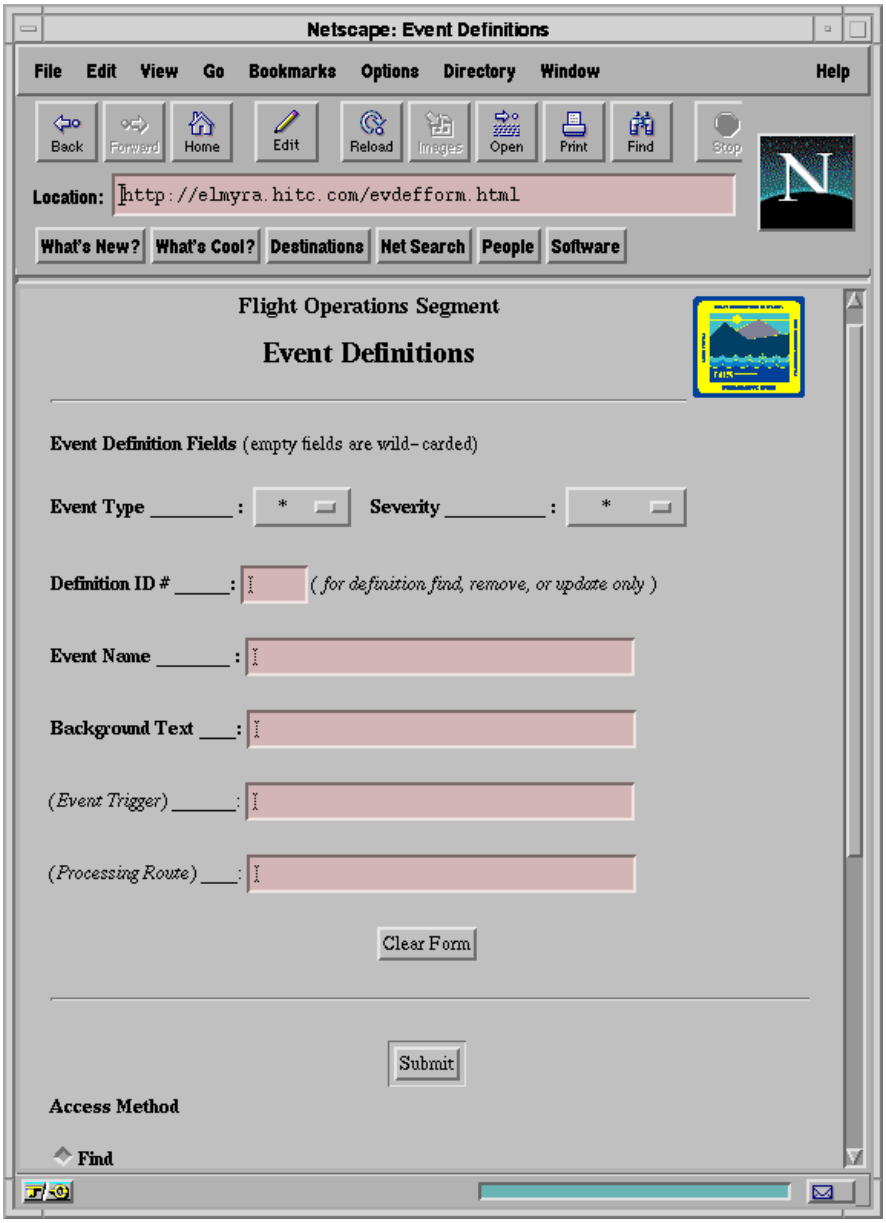

Figure 10.4.4.2-2. Event Definitions Page

8. **Event History**. Open the FOS Event History page (see Figure 10.4.4.2-3) by clicking **Event History** on the FOS Database Events page.

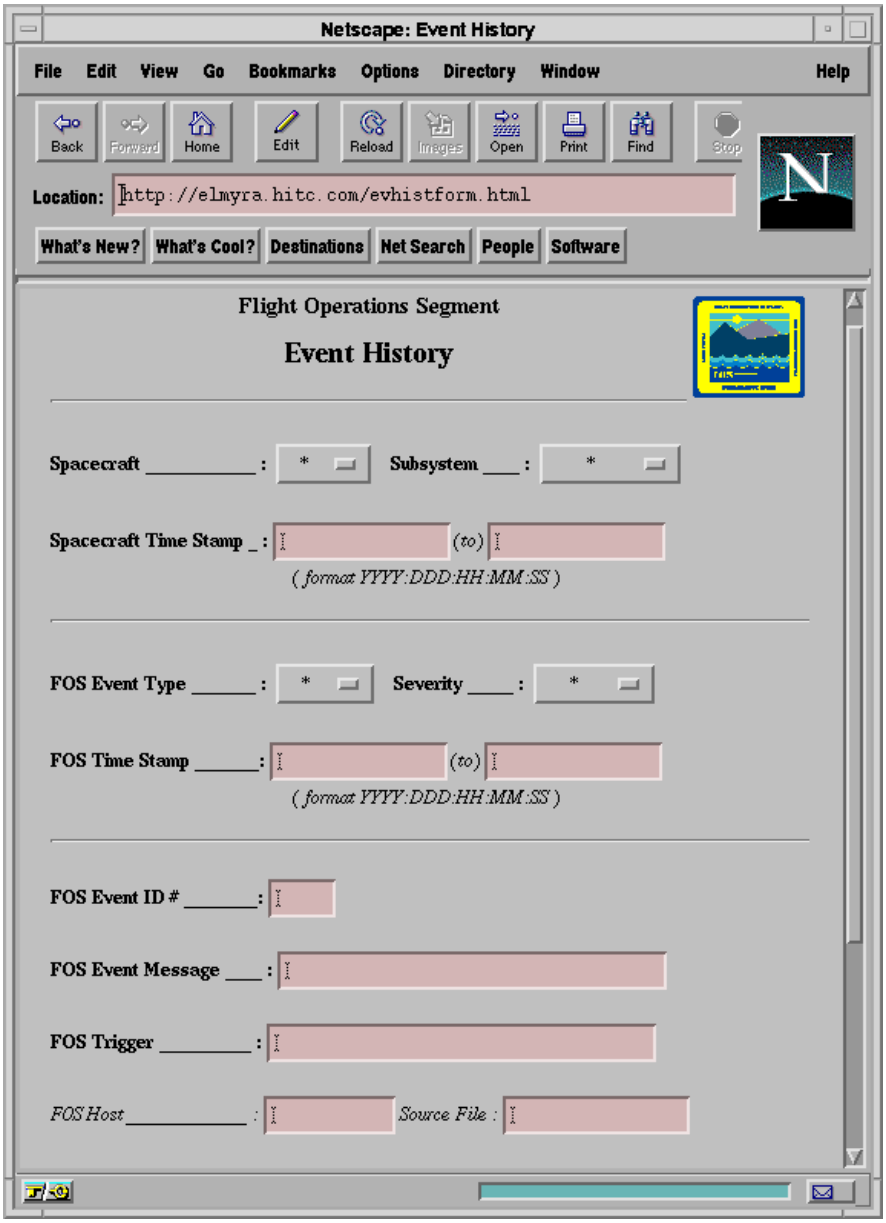

Figure 10.4.4.2-3. FOS Event History Page

The Event History page allows you to retrieve events from the event archive. The following is a list of fields associated with the event history along with a brief field description:

1. **Spacecraft**. Currently, AM1 is the only valid spacecraft.
2. **Subsystem**. The spacecraft or ground subsystem. Examples are Aster, Modis, FDF, and NCC.
3. **Spacecraft Time Stamp**. The start and stop spacecraft times of the event request.

4. **FOS Event Type**: The FOS subsystem generating the event. Valid subsystems are ANA, CMS, CMD, DMS, FUI, PAS, RCM, RMS, TLM, COMMON, TST and SYS.
5. **Severity**. The severity of an event. Valid severities are Fatal, Alarm, Warning, and Informational.
6. **FOS Time Stamp**. The start and stop UTC times of the event request
7. **FOS Event ID#**. The event identifier.
8. **FOS Event Message**. The event message displayed on the Event Display page.
9. **Trigger**. The name of the procedure or executable associated with the event.
10. **FOS Host**. The host name where an event was generated.
11. **Source File**. The source file generating the event.

10.4.4.3 Name Server

Open the Name Server page (see Figure 10.4.4.3-1) by clicking **Name Server** on the Database Access page. The Name Server page allows you to access the Name server database. The following is a list of fields associated with the Name server along with a brief field description:

1. **Entry Id**. The port number of the process.
2. **Host Name**. The host where the process is running.
3. **Service**. The service the process provides.
4. **Service Instance**. The instance of the process on a given host.
5. **Process Name**. The name of the process.
6. **Process Id**. The Unix process ID.
7. **Process State**. The state of the process. Valid states are Active, Backup, and Inactive.
8. **Process Mode**. The mode of the process. Valid modes are Operational, Test, and Training.
9. **Proxy Id**. The class ID of a proxy.
10. **String Id**. The process is associated with this Decom and Parameter Server String. The field is blank if the process is not associated with a string.
11. **Subsystem**. The FOS Subsystem. Valid subsystems are ANA, CMD, CMS, DMS, RCM, RMS, TLM, FUI, and PAS.

12 **Database**. The process is connected to this database. Blank, if not connected to a database.

13. **Spacecraft**. Currently, the only valid spacecraft is AM1.
14. **User**. The user name of the person starting the process.

15. **Role**. The role of the person starting the process.
16. **User String**. A open field. May be used to define a string type and no field is provided.
17. **Channel.** The Channel I = 1, ChannelQ=2, Replay=3, and Config=5.
18. **User Value**. A open field. May be used to define a value type and no field is provided.19. **IPC Type**. The valid IPC types are Point-to-Point and Multicast.

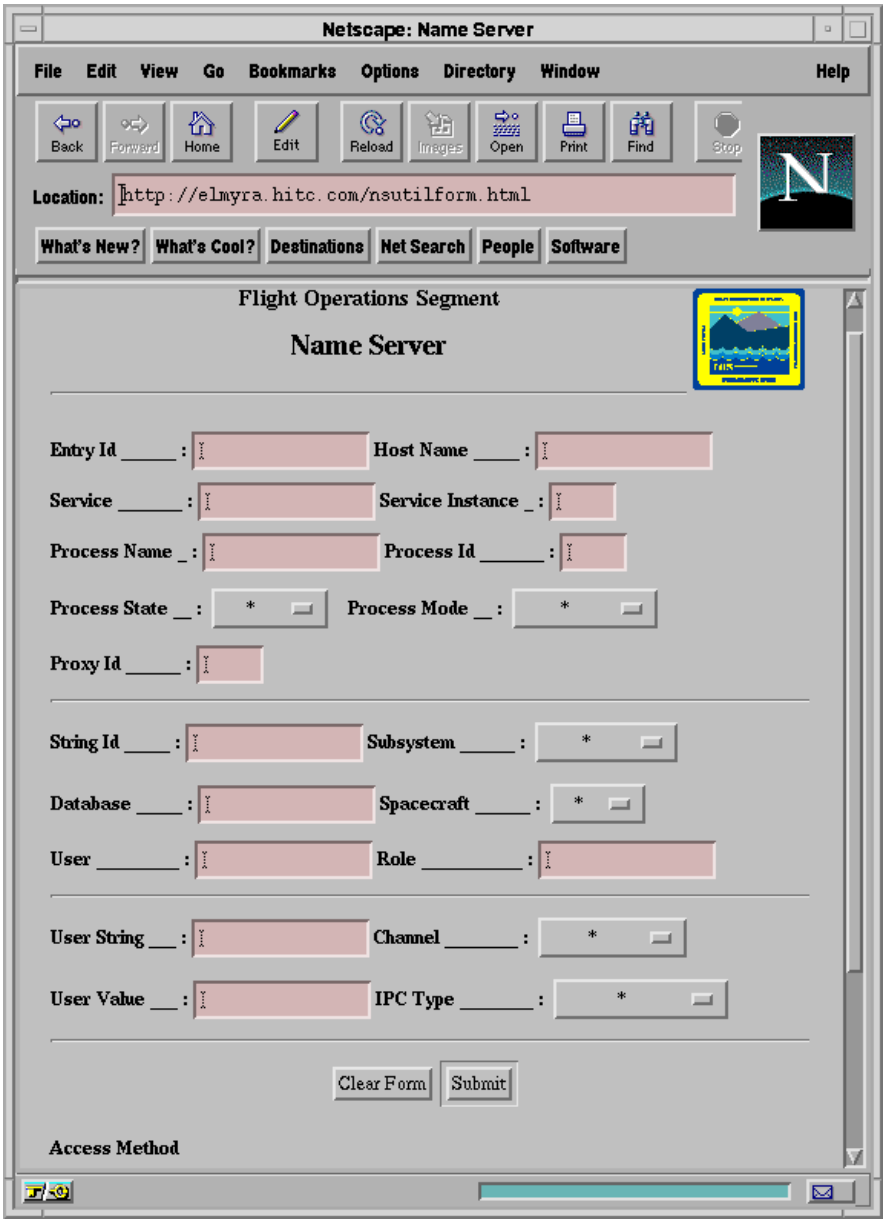

Figure 10.4.4.3-1. Name Server Page